Master's thesis

STORMicroscopy: A Mathematical Analysis

Christoph Florian Schaller

July 2013

Supervisor: Prof. Dr. Frank Noé, FU Berlin

Second assessor: Dr. Jan Schmoranzer, FMP Berlin

Ich versichere hiermit an Eides statt, dass ich die vorliegende Masterarbeit selbstständig und ohne unzulässige fremde Hilfe verfasst habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Berlin, den 02.07.2013

C. Schaller

Contents

1 Introduction **3**

1.1 The diffraction barrier . . . 3
1.2 Imaging and noise . . . 4
1.3 STORM - the basic idea . . . 5
1.4 Motivation and classification of the present thesis . . . 6

2 Fitting algorithms **7**

2.1 Preliminaries and overview . . . 7
2.2 Gaussian mask and full least squares fitting . . . 7
2.3 Numerical integration algorithm . . . 8
2.4 Poissonian background fitting . . . 10
2.5 Fitting a Gaussian background noise . . . 11

3 Uncertainty and precision **14**

3.1 Error estimation . . . 14
3.2 Numerical verification of a systematic error caused by pixelation . . . 15

4 Random STORM images **17**

4.1 Image generation and setup . . . 17
4.2 Error dependencies . . . 17
4.3 Effect of numerical integrations . . . 19
4.4 Quantification of occuring shifts caused by pixelation . . . 20
4.5 Fitting a Poissonian background . . . 22
4.6 Simulation results for Gaussian background fitting . . . 23

5 Processing experimental data **24**

5.1 Available fitting tools . . . 24
5.2 Identifying trajectories . . . 25
5.3 Drift . . . 25

6 xStorm **27**

6.1 Necessity of the development . . . 27
6.2 Input and output . . . 27
6.3 The tagged image file format (TIFF) . . . 27
6.4 Program design . . . 28
6.5 Parallel processing . . . 29

7 Experimental observations **30**

7.1 Noise statistics . . . 30
7.2 Single bead . . . 31
7.3 Multiple beads . . . 32
7.4 Antibody fitting benchmark . . . 34
7.5 Bias due to the fitting method . . . 34

8 Summary and Outlook **36**

8.1 Our results . . . 36
8.2 Future possibilities . . . 36

A Code section **38**
A.1 Simulation environment 38
A.2 xStorm 38

B Assigning the matching background values **39**

Acknowledgment **40**

References **41**

1 Introduction

The first chapter illustrates the difficulties of achieving nanometer resolution in microscopy and gives a brief abstract of the occuring phenomena. Finally the thesis is motivated and outlined.

1.1 The diffraction barrier

In general the resolution of light microscopes is roughly limited by $\lambda/2 \approx 250$ nm, which corresponds to half of the wavelength of the visible spectrum. This is caused by the fact that whenever we observe an object, we actually do not observe a point, but a distribution of photons. Usually the size of that distribution is negligible compared to the resolution. However when it comes to structures in the dimension of nanometers, we start to observe the so called Airy diffraction pattern or Airy disk displayed in Figure 1.1. It is named after the English astronomer George Biddell Airy, who was the first to theoretically treat this phenomenon in the year 1835 [2].

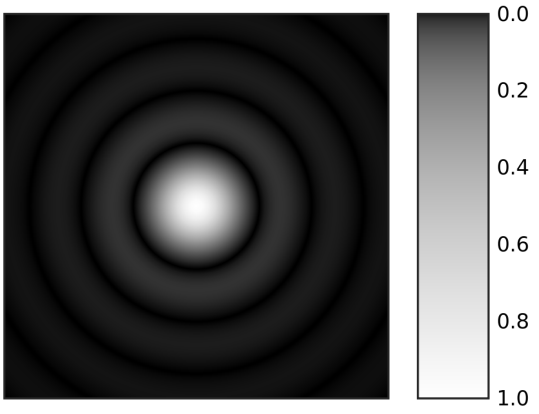

Figure 1.1: Airy disk intensities. [1]

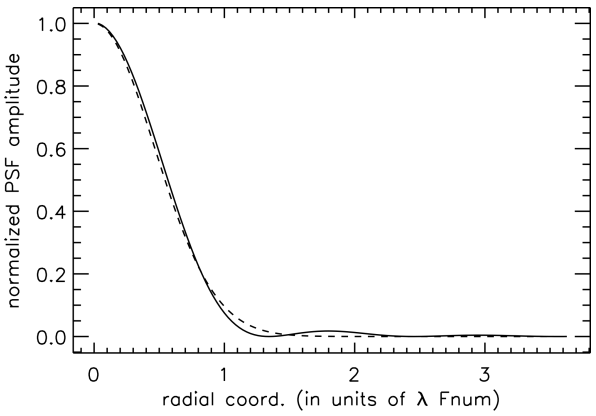

Figure 1.2: Airy disk point spread function (solid) with central ring fit by a Gaussian(dashed). [1]

The pattern consists of several rings, though the central ring contains the major part with 83.8% of the intensity. Its radius can be shown to be

$$r_{Airy} \approx 0.61 \frac{\lambda}{NA}. \tag{1.1}$$

Now the so called Rayleigh criterion declares, that two spots are no longer resolvable if their centers come closer than the radii of their inner rings. Taking the fact into consideration that the best current optical microscopes have numerical apertures NA of up to 1.4 [3], we achieve a lower bound for the optical resolution.

Furthermore the central ring is well-fitted by a Gaussian as displayed in Figure 1.2. As first stated by Thomann et al. in 2002 [4] and proven by Zhang et al. in 2006 [5] the standard deviation for the minimal error in the sense of L_∞ is

$$\sigma \approx 0.21 \frac{\lambda}{NA}.$$

Consequently there are biological structures which can be resolved optically as well as those which we cannot resolve as the scheme in Figure 1.3 points out.

But even if details cannot be resolved, there are multiple approaches to at least locate the center of an object to a much greater precision. One of these is STochastical Optical Reconstruction Microscopy (STORM), which was introduced by Michael J. Rust, Mark Bates and Xiaowei Zhuang in 2006 (cf. [7]) and forms the basis of this thesis.

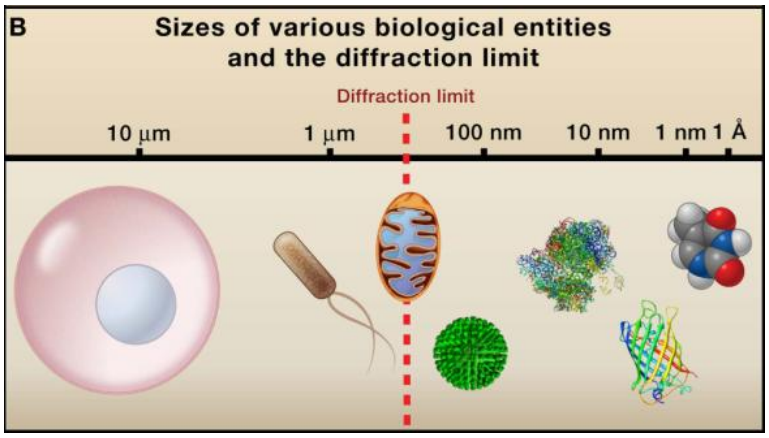

Figure 1.3: The size scale of various biological structures in comparison with the diffraction-limited resolution. [6] (left to right) A mammalian cell, a bacterial cell, a mitochondrion, an influenza virus, a ribosome, the green fluorescent protein, and a small molecule (thymine).

1.2 Imaging and noise

Wanting to work with these distributions of photons one would like to localize single photons as precise as possible. However it is only possible to collect photons in so called pixels and measure the current they cause at a photoelectric cell. Here we got our first crucial source of noise: As we collect photons within an area, we do not know from which exact location the photon originated. This uncertainty is called “pixelation noise”.

Additionally the measured current does not indicate the exact number of photons within the area as there are effects like dark current or incoming photons generating different numbers of electrons. Those inaccuracies are regarded as global influences and therefore their impact is summarized under the term “background noise” together with effects such as out-of-focus fluorescence and readout errors.

Finally we do not always observe a perfect distribution but a fixed number of localizations generated by the distribution. The influence of that factor is called “photon shot noise”, but becomes negligible for large enough photon numbers.

Taking into account that pixel sizes of about 100 nm are common, the imaging process as depicted in Figure 1.4 seems pretty rough. Nevertheless the distributions remain observable and thus we can determine their most likely centers.

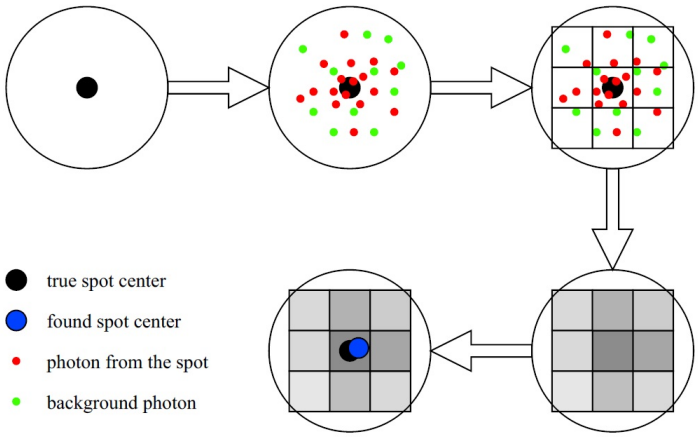

Figure 1.4: The imaging process is aggravated by different sources of noise and pixelation.

1.3 STORM - the basic idea

The basic idea is presented in principle in the adjacent scheme in Figure 1.5. At first photoswitchable fluorophores are attached to specific molecules, e.g. nucleic acids or proteins, in an immobilized sample. Then one (optically resolvable) subset is activated by laser excitation of a specific wavelength. Thereafter the activated fluorophores emit photons while imaging occurs. That way one obtains a coarse matrix for every frame, which contains the number of collected photons within every pixel. Using these one is now able to reconstruct the spot centers utilizing a so called "fitting algorithm". After waiting for the activated fluorophores to go back into a dark state, one can repeat the described process several times to obtain a STORM image until most of the fluorophores were excited at least once.

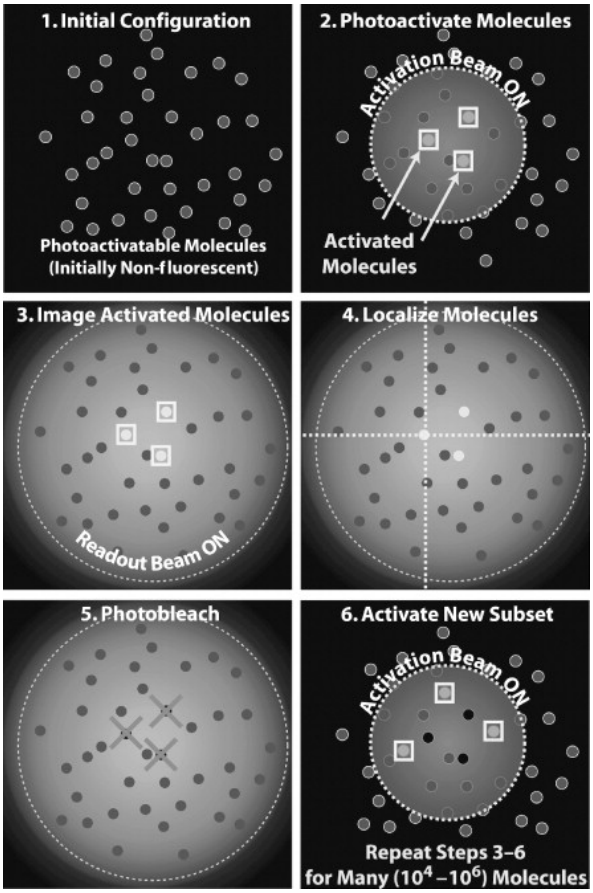

Figure 1.5: The STORM imaging process. [8]

As the cautious reader might have observed there are several preconditions to be satisfied. On the one hand we would like to have well-separated objects to easily distinguish them from each other, on the other hand we need to be able to label only specific subsets in a discriminable way. Furthermore if our objects are too large themselves, reducing them to one point is not very meaningful, thus we assume them small enough to be considered punctate.

The following Figure 1.6 depicts the improved resolution due to STORM compared to immunofluorescence microscopy.

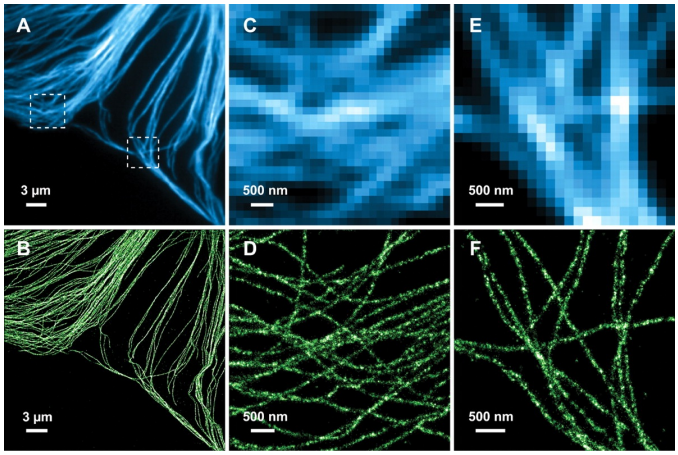

Figure 1.6: STORM imaging of microtubules in a mammalian cell. [9] (A) Conventional immunofluorescence image in a large area. (B) STORM image of the same area. (C and E) Conventional and (D and F) STORM images corresponding to the boxed regions in (A).

1.4 Motivation and classification of the present thesis

Several authors claim to achieve "nanometer precision" using STORM. This coincides with the theoretical predictions yielding that for a "sufficient" number of photons in a spot we should be able to locate its center arbitrarily accurate. However for simulated data as well as microscopic measurements we observe lower resolutions as desired, probably resulting from approximations in the used models or algorithms. Therefore it seems necessary to have a closer look at all steps of the fitting process to detect error sources and look for possible improvements.

2 Fitting algorithms

As a starting point for the upcoming research the predominant theoretical status is recapitulated. Then according to our findings we will try to improve the currently used fitting algorithms.

2.1 Preliminaries and overview

Let some experimental result, i.e. a large matrix with photon counts for every pixel, be given. First of all we have to locate spots within this matrix roughly, before we can commence with fitting the spot centers. Assuming that there are no overlapping spots, this can be done using the Algorithm 2.1.

Algorithm 2.1 Simple spot recognition.

- Find pixel values above some threshold (e.g. "8 standard deviations away from the mean" [10] of the intensity value distribution).
- Look for local maxima within connected regions of such pixels.
- Cut out surrounding regions according to the known size of a potential spot.
- Average the remaining cell values of the frame to obtain the background mean.
- Subtract the background mean from the spots.
- Fit the spot centers.

Unfortunately the fit of the background noise is only treated in a very simple manner here. As long as its (standard) deviation is "sufficiently" small this may be adequate, but a localized background fit might be more precise. Nonetheless this implies a higher computational effort, so first we use the same approach as previous authors and analyze its behaviour.

In [11] Cheezum et al. applied four commonly used single particle tracking algorithms under realistic conditions. In comparison to the centroid algorithm, cross-correlation and the sum-absolute difference (SAD) method, a direct Gaussian fit to the intensity distribution turns out to be the best choice for point sources in terms of robustness and precision.

2.2 Gaussian mask and full least squares fitting

Now assume a matrix of observed photon counts (without background noise) for each pixel within a possible spot location is given and denote it S_{ij}, where $(i,\ j)$ defines the location of the pixel center with respect to a local coordinate grid. Moreover indicate with $(x_0,\ y_0)$ the unknown spot center and the likewise unknown total number of photons within the spot with N. Now let $p_G(i,j)$ be the probability density function of a (normalized) Gaussian distribution centered in $(x_0,\ y_0)$ with known (it can be calculated from the emission wavelength) standard deviation σ, i.e.

$$p_G(i,j) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{(i-x_0)^2}{2\sigma^2} - \frac{(j-y_0)^2}{2\sigma^2}\right).$$

In the following we want to approximate the center of the spot by fitting S_{ij} with a Gaussian curve given by $G_{ij} := N \cdot p_G(i,j)$, which is a pixelated approximation of the expected number of photons. Indeed for every pixel $I = [i-\frac{1}{2}, i+\frac{1}{2}] \times [j-\frac{1}{2}, j+\frac{1}{2}]$ we use $\int_I\ p_G(x,y)dA \approx p_G(i,j) \cdot A_I = p_G(i,j)$. Thus we require a small enough pixel size to justify this approximation.

For fitting with a Gaussian in the next step a maximum likelihood estimation is done by the least squares approach, i.e. we want to minimize $\chi^2 = \sum_{i,\ j} \frac{(S_{ij}-G_{ij})^2}{\sigma_{ij}^2}$. Here σ_{ij} denotes the local uncertainty of the pixel values, which we assume to be constant across one spot. We know that the

x- and y-direction of a two-dimensional Gaussian distribution are independent. Thus we can consider minimization in direction of x_0 and y_0 separately. Let us start with the x-coordinate.

For every minimum of χ^2:

$$0 = \frac{d}{dx_0} \sum (S_{ij} - G_{ij})^2$$

$$\iff 0 = \sum 2(S_{ij} - G_{ij}) \frac{d}{dx_0} G_{ij}$$

$$\iff 0 = \sum (S_{ij} - G_{ij}) \frac{(i - x_0)}{\sigma^2} G_{ij}$$

$$\iff 0 = \sum S_{ij} G_{ij} (i - x_0) - \sum G_{ij}^2 (i - x_0) \tag{2.1}$$

If we now use the approximation $\sum G_{ij}^2 (i - x_0) \approx 0$ due to odd symmetry as $G_{x_0+k,\, j} \approx G_{x_0-k,\, j}$ and $(x_0 + k - x_0) = -(x_0 - k - x_0)$, we obtain the Gaussian mask algorithm described by Thompson et al. in [10]. Ultimately, the remaining equation $0 = \sum S_{ij} G_{ij} (i - x_0)$ yields the iteration

$$x_0 = \frac{\sum i S_{ij} G_{ij}}{\sum S_{ij} G_{ij}}.$$

In particular, this equation does not depend on N anymore as it cancels after plugging in $G_{ij} = N \cdot p_G(i, j)$. Then p_G depends only on $(x_0,\ y_0)$, while the S_{ij} are known. Analogously we obtain an iteration formula for y_0 resulting in the following parallel iteration for both coordinates

$$x_0 = \frac{\sum i S_{ij} p_G(i, j)}{\sum S_{ij} p_G(i, j)},\ y_0 = \frac{\sum j S_{ij} p_G(i, j)}{\sum S_{ij} p_G(i, j)}. \tag{2.2}$$

However if we do not neglect the second term we can achieve higher accuracy at the cost of some extra computational effort as then (2.1) ensues the iteration

$$x_0 = \frac{\sum i (S_{ij} - G_{ij}) G_{ij}}{\sum (S_{ij} - G_{ij}) G_{ij}}. \tag{2.3}$$

Again we come up with a similar equation for y_0, but this time our iteration is not independent of N. Thus here we have to fit the total number of photons at the same time. This can be achieved by adding up the total photon count within the spot with respect to the current $(x_0,\ y_0)$ or more accurate by using the equation

$$N = \frac{\sum S_{ij} p_G(i, j)}{\sum p_G(i, j)^2}, \tag{2.4}$$

where the pixel counts are weighted with the probability to hit the considered pixel. The parallel iteration of the position equations from (2.3) with (2.4) is mostly called full least squares fitting.

Nonetheless when performing the Gaussian mask algorithm, we only need to calculate N once from (2.4) in the very end to gain an approximation of the total number of photons in the spot.

2.3 Numerical integration algorithm

When having a look at the currently used algorithms one observes that they all approximate integrals of the point spread function by assuming a constant value inside of every pixel which is equal to the one at its center. Right now we are trying to avoid this approximation.

Recall the notations from Chapter 2.2, i.e. S_{ij} is the matrix of photon counts, σ the standard deviation, N the unknown number of photons and $(x_0,\ y_0)$ the spot center, that we want to approximate. Now instead of using a pixelated Gaussian we want to use the exact Gaussian distribution, which was called p_G in the two-dimensional case.

Once more we can limit the fitting to the one-dimensional case as all occuring distributions are rotationally symmetric. Hence we use $C_i = \sum_j S_{ij}$ as the observation data and fit with $G_i = N\int_{i-\frac{1}{2}}^{i+\frac{1}{2}} p_{1D}(x)dx$, where p_{1D} denotes the one dimensional Gaussian distribution centered in x_0 with standard deviation σ, i.e. $p_{1D}(x) = \frac{1}{\sqrt{2\pi}\sigma}\exp(-\frac{(x-x_0)^2}{2\sigma^2})$.

We repeat the least squares approach

$$0 = \frac{d}{dx_0}(\sum_i (C_i - G_i)^2)$$

$$\iff 0 = \sum 2(C_i - G_i)\frac{d}{dx_0}G_i,$$

but now requiring to calculate $\frac{d}{dx_0}G_i = N\frac{d}{dx_0}\int_{i-\frac{1}{2}}^{i+\frac{1}{2}} p_{1D}(x)dx$.

Luckily we can switch integral and differentiation here as $[i-\frac{1}{2}, i+\frac{1}{2}]$ is finite, $p_{1D}(x)$ is continuous and $\frac{d}{dx_0}p_{1D}(x)$ exists and is continuous, too. Therefore we obtain $\frac{d}{dx_0}G_i = N\int_{i-\frac{1}{2}}^{i+\frac{1}{2}} \frac{(x-x_0)}{\sigma^2}p_{1D}(x)dx$ and thus $0 = \sum(C_i - G_i)\int_{i-\frac{1}{2}}^{i+\frac{1}{2}}(x-x_0)p_{1D}(x)dx$.

Knowing that $e(x) = \frac{1}{2}\text{erf}(\frac{x-x_0}{\sigma\sqrt{2}})$ satisfies $e'(x) = p_{1D}(x)$, plugging in yields $G_i = N\left(e(i+\frac{1}{2}) - e(i-\frac{1}{2})\right)$. For simplicity we denote $e_{i\pm} = e(i \pm \frac{1}{2})$. Furthermore we can integrate

$$\int_{i-\frac{1}{2}}^{i+\frac{1}{2}} (x-x_0)p_{1D}(x)dx = \left[-\sigma^2 p_{1D}(x)\right]_{i-\frac{1}{2}}^{i+\frac{1}{2}}.$$

For our least squares problem follows

$$0 = \sum(C_i - Ne_{i+} + Ne_{i-})\sigma^2[p_{1D}(i+\frac{1}{2}) - p_{1D}(i-\frac{1}{2})]$$

$$\iff 0 = \underbrace{\sum(C_i - Ne_{i+} + Ne_{i-})\left(\exp(-\frac{(i+\frac{1}{2}-x_0)^2}{2\sigma^2}) - \exp(-\frac{(i-\frac{1}{2}-x_0)^2}{2\sigma^2})\right)}_{=:f(x_0)}.$$

Now we "just" need to solve this nonlinear equation. If we do not want to approximate, an application of Newton's method for $f(x_0)$ started in the pixel center of the local maximum should suffice. In order to apply the iteration $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$ we need to know $f'(x_0)$, too. As $e_{i\pm} = e(i \pm \frac{1}{2})$ depends on x_0, the product rule yields

$$f'(x_0) = \sum(Np_{1D}(i+\frac{1}{2}) - Np_{1D}(i-\frac{1}{2}))\left(\exp(-\frac{(i+\frac{1}{2}-x_0)^2}{2\sigma^2}) - \exp(-\frac{(i-\frac{1}{2}-x_0)^2}{2\sigma^2})\right)$$

$$+\sum(C_i - Ne_{i+} + Ne_{i-})\left(\frac{i+\frac{1}{2}-x_0}{\sigma^2}\exp(-\frac{(i+\frac{1}{2}-x_0)^2}{2\sigma^2}) - \frac{i-\frac{1}{2}-x_0}{\sigma^2}\exp(-\frac{(i-\frac{1}{2}-x_0)^2}{2\sigma^2})\right).$$

By denoting

$$p_{i\pm}(x_n) := \exp(-\frac{(i\pm\frac{1}{2}-x_n)^2}{2\sigma^2}) = \sqrt{2\pi}\sigma p_{1D}(i\pm\frac{1}{2})$$

and generalizing

$$e_{i\pm}(x_n) := \frac{1}{2}\text{erf}(\frac{i\pm\frac{1}{2}-x_n}{\sigma\sqrt{2}})$$

we can write

$$f'(x_n) = \sum \frac{N}{\sqrt{2\pi}\sigma}(p_{i+} - p_{i-})^2 + \sum (C_i - Ne_{i+} + Ne_{i-})\frac{1}{\sigma^2}\left((i + \frac{1}{2} - x_n)p_{i+} - (i - \frac{1}{2} - x_n)p_{i-}\right)$$

and

$$f(x_n) = \sum (C_i - Ne_{i+} + Ne_{i-})(p_{i+} - p_{i-}).$$

Finally we need some way of calculating the values $e_{i\pm} = \frac{1}{2}\text{erf}(\frac{i\pm\frac{1}{2}-x_0}{\sigma\sqrt{2}})$. This can be done by one of several approximations for the error function according to the needed accuracy. In MATLAB the build-in function $\text{erf}(x)$ performs efficiently with relative errors of order 10^{-19} as it is an implementation of the algorithm given by W. Cody in [12]. As we obtain an analogous relation $g(y_n)$ in y-direction one can iterate

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)},\ y_{n+1} = y_n - \frac{g(y_n)}{g'(y_n)},$$

starting off with $(x_0,\ y_0)$ located in the pixel center.

But as $f(x_n)$, $f'(x_n)$ and $g(y_n)$, $g'(y_n)$ are still N-dependant, this needs to be done in turn with the weighted sum

$$N = \frac{\sum S_{ij}P_{ij}(x_n, y_n)}{\sum P_{ij}(x_n, y_n)^2},$$

where $P_{ij}(x_n, y_n)$ is the probability to hit pixel $(i,\ j)$ from center $(x_n,\ y_n)$ and consequently

$$P_{ij} = \int_{i-\frac{1}{2}}^{i+\frac{1}{2}}\int_{j-\frac{1}{2}}^{j+\frac{1}{2}} p_G(x,y)dy\ dx = \frac{1}{4}\left[\text{erf}(\frac{x-x_n}{\sigma\sqrt{2}})\right]_{i-\frac{1}{2}}^{i+\frac{1}{2}}\left[\text{erf}(\frac{y-y_n}{\sigma\sqrt{2}})\right]_{j-\frac{1}{2}}^{j+\frac{1}{2}} = (e_{i+,x} - e_{i-,x})(e_{i+,y} - e_{i-,y}).$$

Nonetheless the required values of the error function are the same ones required for the iterations of the pixel center and thus only need to be calculated once.

All in all this provides a method using no approximations apart from the calculation of the error functions which can be done to whatever precision needed.

2.4 Poissonian background fitting

After avoiding approximations in the fitting algorithm itself we now want to have a closer look at the background noise and how it is treated. We recall that background noise was estimated in a rather simple fashion so far. Obviously subtracting a constant value from every pixel does not represent the reality as the background noise is a random process following a certain distribution. As stated in [10] and universally accepted, the background noise can be seen as a Poisson process and therefore every pixel value should stem from the same Poisson distribution. Thus we want to include fitting a noise value for every pixel within a spot.

Assume a matrix K_{ij} of observed photon counts including background noise is given. Now we introduce a matrix b_{ij}, which is meant to contain the number of photons most probably steming from the background. Finally G_{ij} denotes the matrix of the currently fitted Gaussian distribution and b the average background noise value.

Thence the probability of an observation is

$$P_{ij} = \underbrace{\frac{G_{ij}^{K_{ij}-b_{ij}}e^{-G_{ij}}}{(K_{ij}-b_{ij})!}}_{P_{PSF}} \cdot \underbrace{\frac{b^{b_{ij}}e^{-b}}{b_{ij}!}}_{P_{back}}.$$

Here P_{PSF} is the probability of observing $K_{ij} - b_{ij}$ photons from the distribution G_{ij} and P_{back}

the probability to observe b_{ij} background photons if these have an average of b. Both processes are considered to be Poissonian as there is a certain probability of success and a fixed number of independently drawn samples.

Now we can interpret this as a function of b_{ij}, i.e.

$$\frac{P_{ij}(b_{ij})}{c} = \frac{G_{ij}^{-b_{ij}} b^{b_{ij}}}{b_{ij}!(K_{ij} - b_{ij})!},$$

where $c \in \mathbb{R}$ is independent of b_{ij}. Consequently maximizing the right hand side with respect to b_{ij} gives us the value maximizing P_{ij}.

Utilizing this idea, we introduce the following Algorithm 2.2 as an improved way of treating the background noise.

Algorithm 2.2 Improved background fit.

- Start with a matrix of photon counts K_{ij}.
- Search for local maxima.
- Cut out surrounding regions according to the potential spots.
- Initialize $b_{ij,0} = b$, where b is the average of the remaining cells, i.e. the background mean.
- Repeat the following iteration steps up to a fixed accuracy.
  - Calculate the matrix of photon counts $S_{ij,n} = K_{ij} - b_{ij,n}$.
  - Do one iteration step for $(x_n,\ y_n)$ and N using a fitting algorithm.
  - Maximize $P_{ij}(b_{ij})$ for all $(i,\ j)$.
  - Calculate b as the average of all b_{ij}.

The only remaing question is how to maximize $P_{ij}(b_{ij})$. We cannot simply use the first derivative, thus we are looking for a maximizer $b_{ij} \in \mathbb{N}$. As we know the average values of the Poisson distributions, P_{PSF} is maximized by $K_{ij} - G_{ij}$ and P_{back} by b. Accordingly we can systematically compare the values of P_{ij} in the interval and find a local maximizer.

2.5 Fitting a Gaussian background noise

2.5.1 The approach

As we will establish in Chapter 7.1, the background noise can be well-estimated by a Gaussian distribution. We want to try to include this into our fitting algorithm to obtain even better results than before. In the previous algorithms we always subtracted the average background noise from every pixel and used the result as input for our fitting algorithm. This is reasonable for a Poissonian (assumed) as well as a Gaussian (observed) as then the average background noise is zero for every pixel. However this increases the uncertainty of every pixel value according to the standard deviation of the underlying background distribution. Thus the variations are much higher for the observed Gaussian, which has a significantly larger standard deviation than the corresponding Poissonian for the same mean value.

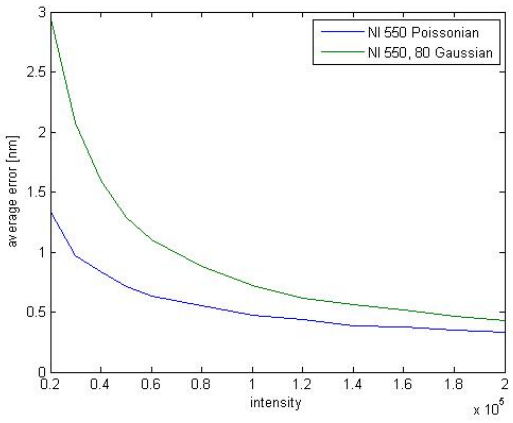

Figure 2.1: Average error according to intensity, pixel size 100 nm, spot diameter 500 nm.

To emphasize this we generate 1000 STORM images (for different intensities) for both cases, fit them with our numerical integration algorithm and plot the average errors in Figure 2.1. We use an average intensity of 550 per pixel respectively and a standard deviation of 80 for the Gaussian according to our noise statistics. In consequence of the higher standard deviation we observe a significantly larger average fitting error caused by the increased variations of the individual pixel values. This motivates the attempt to include our knowledge of the background distribution into a fitting algorithm to achieve lower average errors.

So far we minimized $\chi^2 = \sum(\underbrace{O_{ij} - b}_{S_{ij}} - G_{ij})^2$, where O_{ij} denoted the original values, b the mean background and G_{ij} the distribution of our current fit. Instead we now want to work with $\chi^2 = (O_{ij} - b_{ij} - G_{ij})^2$, where b_{ij} denotes a so called "local" background fit. Unfortunately it is not possible to minimize with respect to all free parameters $(x,\ y,\ b_{ij})$, because the background fit b_{ij} is position-dependant in an analytically unknown way. Thus we want to generate a pool of discrete Gaussian distributed background values and assign them to the individual pixels.

Previously the fitting of Poissonian background and position in turn did not result in significant improvements as seen in Chapter 2.4. The reason for that may be that separate fitting damps the iteration steps in direction of the initial position as the difference of observations and old fit is treated as background. Therefore we will analyze whether taking the result of the common numerical integration algorithm as an initial iterate for the new algorithm improves its quality. This seems justified as we will have an unbiased starting point for our background fitting, which will hopefully shift it further on towards the true center, even if it does not arrive there due to damping. Still the result would be an improvement of the best possible fit that was developed until now.

2.5.2 The discrete approximation of a Gaussian

As mentioned before, we will need a pool of background values distributed according to a known Gaussian. Here we describe how to obtain the most probable distribution of such discrete values.

First of all, assume we know the average $m \in \mathbb{R}$ and the standard deviation $s \in \mathbb{R}$ of the underlying background, because we can estimate them from the pixels which were not assigned to a spot. Then the probability to observe a fixed number $k \in \mathbb{N}$ of background photons is approximately $P(k) = \int_{k-\frac{1}{2}}^{k+\frac{1}{2}} \frac{1}{\sqrt{2\pi}s}\exp(-\frac{(z-m)}{2s^2})dz$, a number which we can easily calculate. Now we can inductively find the most probable "next" photon number by basic stochastics as the probality to observe a specific distribution of n pixel values is a multiset permutation and therefore given by

$$P = \frac{n!}{a_1! \cdot a_2! \cdot \ldots \cdot a_n!} P(k_1)^{a_1} \cdot \ldots \cdot P(k_n)^{a_n},$$

where a_i denotes the number of occurences of the background noise value k_i.

Altogether this motivates the following inductive Algorithm 2.3 to obtain the desired distribution.

Algorithm 2.3 Generating the most probable discrete Gaussian distribution.

- The most probable distribution for $k = 1$ is of course $D_1 = \{m\}$.
- Now for $k = n + 1$, we can obtain D_{n+1} from D_n.
  - Denote the maximal and minimal $i \in \mathbb{Z}$ for which $m + i$ occurs in D_n by t_{min} and t_{max}.
  - Calculate $z_i = \frac{P(m+i)}{a_{m+i}+1}$, where a_{m+i} is the number of occurences of $m + i$ in D_n, for all $i \in \{t_{min} - 1,\ ...,\ t_{max} + 1\}$.
  - Denote the $i \in \mathbb{Z}$ for which z_i is maximal by t and obtain $D_{n+1} = D_n \cup \{t\}$.

2.5.3 The algorithm itself

We recall our algorithm for one coordinate from Chapter 2.3, which can be summarized by the following equations:

$$f(x_n) = \sum (C_i - Ne_{i+} + Ne_{i-})(p_{i+} - p_{i-}),$$

$$f'(x_n) = \sum \frac{N}{\sqrt{2\pi}\sigma}(p_{i+} - p_{i-})^2 + \sum (C_i - Ne_{i+} + Ne_{i-})\frac{1}{\sigma^2}\left((i + \frac{1}{2} - x_n)p_{i+} - (i - \frac{1}{2} - x_n)p_{i-}\right) \text{ and}$$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

Here we denoted the number of photons by N, the standard deviation by σ, the 1D-observations by $C_i = \sum_j S_{ij}$ and furthermore

$$p_{i\pm}(x_n) := \exp(-\frac{(i \pm \frac{1}{2} - x_n)^2}{2\sigma^2}) = \sqrt{2\pi}\sigma p_{1D}(i \pm \frac{1}{2}) \text{ and}$$

$$e_{i\pm}(x_n) := \frac{1}{2}\text{erf}(\frac{i \pm \frac{1}{2} - x_n}{\sigma\sqrt{2}}).$$

The point that we want to tackle now is $C_i = \sum_j S_{ij} = \sum_j (O_{ij} - b)$, where b was the average background noise.

Let us assume we have $M \times M$ pixels that contain a spot. Then the background noise in one column (or analogously row) is the sum of M background values. We recall our notations and algorithms from the previous Chapter 2.5.2. Thus the number of background photons in one column is normally distributed with mean $M \cdot m$ and standard deviation $\sqrt{M} \cdot s$ as the sum of normal distributions is again normally distributed with the means and variances summed. Now we can use Algorithm 2.3 to calculate the M most probable background values D_M. Finally we need to assign these values to the columns.

We recall that we minimize $\chi^2 = \sum_{i=1}^{M}(O_i - b_i - G_i)^2 = \sum_{i=1}^{M}(O_i - b_i - Ne_{i+} + Ne_{i-})^2$, where we denote a columnwise background fit by b_i. Now for the actual iterate x_n we can calculate the differences $(O_i - Ne_{i+} + Ne_{i-})$ and assign the values $b_i \in D_M$ such that χ^2 is minimal. This is simply done by ordering the differences and background values respectively and grouping the ones in the same places (cf. Appendix B for the proof).

3 Uncertainty and precision

Before the algorithms will be tested, we assess the inherent inaccuracies in the fitting process.

3.1 Error estimation

In [10] Thompson et al. furthermore provided the single error estimation for the localization precision of least squares fitting. This is done by splitting up the problem into the two extreme cases of few or many photons (compared to the background noise) being present.

We start with the so called photon shot noise-limited case, where we assume that no background noise is present or it is negligible compared to the occuring photon numbers. As long as no pixelation occurs the error can be estimated by the common statistical formula $\langle(\triangle x)^2\rangle = \frac{\mathrm{Var}(x)}{N} = \frac{\sigma^2}{N}$.

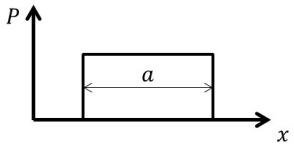

Figure 3.1: Top-hat distribution of size a.

Secondary to σ denoting the standard deviation, we use a for the pixel size and thus $\frac{a^2}{12}$ is the variance of a top-hat distribution of size a, which can be seen in Figure 3.1. Therefore adding the pixelation noise results in

$$\langle(\triangle x)^2\rangle = \frac{\sigma^2 + \frac{a^2}{12}}{N}.$$

The second case, where we assume that no pixelation occurs, is a little more complex. We recall that we minimized $\chi^2 = \sum_{i,j} \frac{(S_{ij}-G_{ij})^2}{\psi_{ij}^2}$, where the observations were denoted as S_{ij}, our fit as G_{ij} and ψ_{ij} was the local uncertainty. Anyhow we can limit ourselves to the one-dimensional case again as a 2D Gaussian equals a pair of independent 1D Gaussians in each coordinate direction. Thus we only have to consider $\chi^2 = \sum_k \frac{(S_k-G_k)^2}{b^2}$ and furthermore use $\psi_k = b\ \forall k$, because all uncertainty is background noise with standard deviation b.

Now we apply a Taylor approximation for G_k, i.e. $G_k(x) = G_k(x_0) + \underbrace{(x-x_0)}_{\triangle x} G_k'(x_0) + O((\triangle x)^2)$.

Additionally we denote $\triangle S_k = G_k(x_0) - S_k$.

$$0 = \frac{d}{dx}\chi^2 = \sum_k 2\frac{(S_k - G_k(x))}{b^2} G_k'(x)$$

$$\iff 0 = \sum_k \left(-\triangle S_k - \triangle x \cdot G_k'(x_0)\right) G_k'(x_0) + O((\triangle x)^2)$$

$$\iff 0 = \sum_k \triangle S_k G_k'(x_0) + \sum_k G_k'(x_0)^2 \triangle x + O((\triangle x)^2)$$

$$\implies \triangle x \approx -\frac{\sum_k \triangle S_k G_k'(x_0)}{\sum_k G_k'(x_0)^2} \tag{3.1}$$

We know that $G_k'(x_0)$ are constants and $\langle(\triangle S_k)^2\rangle = \mathrm{Var}(S_k) = b^2$. Moreover $\langle\triangle S_k\rangle \approx 0$ holds as the values of S_k are symmetrically distributed with respect to $\langle S_k\rangle \approx G_k(x_0)$. Plugging this in results in the following calculations.

$$\langle\left(\sum_k \triangle S_k G_k'(x_0)\right)^2\rangle = \sum_k \langle(\triangle S_k)^2\rangle G_k'(x_0)^2 = \sum_k b^2 G_k'(x_0)^2$$

$$\underset{(3.1)}{\implies} \langle(\triangle x)^2\rangle = \frac{b^2}{\sum_k G_k'(x_0)^2}$$

From $G_k(x) = \frac{N}{\sqrt{2\pi}\sigma}\exp(-\frac{(k-x)^2}{2\sigma^2})$ we derive $G_k'(x) = \frac{N\cdot(k-x)}{\sqrt{2\pi}\sigma^3}\exp(-\frac{(k-x)^2}{2\sigma^2})$ and consequently $G_k'(x)^2 = \frac{N^2\cdot(k-x)^2}{2\pi\sigma^6}\exp(-\frac{(k-x)^2}{\sigma^2})$. Finally we use $\sum_k G_k'(x_0)^2 \approx a\int G_k'(x_0)^2 dk$.

$$\int_{-\infty}^{\infty} \frac{(k-x)^2}{\sigma^2} \exp\left(-\frac{(k-x)^2}{\sigma^2}\right) dk = \frac{\sigma\sqrt{\pi}}{2} \tag{3.2}$$

$$\Longrightarrow a \cdot \int_{-\infty}^{\infty} G'_k(x_0)^2 dk = a \cdot \frac{N^2}{2\pi\sigma^4} \cdot \frac{\sigma\sqrt{\pi}}{2} = \frac{aN^2}{4\sqrt{\pi}\sigma^3}$$

$$\Longrightarrow \langle(\triangle x)^2\rangle = \frac{4b^2\sqrt{\pi}\sigma^3}{aN^2}$$

Similar calculations for the two dimensional case yield $\langle(\triangle x)^2\rangle = \frac{8b^2\pi\sigma^4}{a^2N^2}$. The main difference is a double integral in (3.2) resulting in another factor of $\frac{\sigma\sqrt{\pi}}{2}$.

At last we approximate the error between the two extreme cases by the sum of both estimations, resulting in

$$\langle(\triangle x)^2\rangle \approx \frac{\sigma^2 + a^2/12}{N} + \frac{8b^2\pi\sigma^4}{a^2N^2}. \tag{3.3}$$

This curve has a transition point, meaning that up to some critical number of photons the error behaves like $1/N$, but for large enough photon numbers decays with $1/\sqrt{N}$. It can be found where both terms of the sum equal each other, i.e. $\frac{\sigma^2+\frac{a^2}{12}}{N_t} = \frac{4b^2\sqrt{\pi}\sigma^3}{aN_t^2}$, which is true for

$$N_t = \frac{4\sqrt{\pi}\sigma^3 b^2}{a(\sigma^2 + a^2/12)}.$$

3.2 Numerical verification of a systematic error caused by pixelation

In our simulations we will observe an improvement of the average error due to numerical integrations. To support these results we now want to show the existence of a systematical error for fitting with a pixelated Gaussian. This means that not only the average error is smaller for numerical integrations, but the mean center is unbiased, too.

Treating the error caused by pixelation analytically is hardly possible because of the occuring integrals of the Gaussian probability density function. Still we can easily show the existence of a systematic error due to pixelation as follows.

We assume to observe a perfect Gaussian distribution with no background noise, i.e. real photon values for each pixel. If we used our numerical integration algorithm from Chapter 2.3, we would of course be able to refit the center exactly as we fit with the very same distribution. However when using the pixelated approach, i.e. setting the value of the Gaussian at the center as constant above the whole pixel (cf. Chapter 2.2), this is not necessarily true any longer.

Thus we systematically perform least squares fits of a pixelated Gaussian to a Gaussian on a grid with equally distributed spot centers within one pixel, which can be seen in Figure 3.2. The observed effects are easier to understand remembering that a 2D Gaussian is the combination of 1D Gaussians in x- and y- direction. Now as long as the center is located at 0, 0.5 or 1 for one coordinate, the Gaussian distribution is symmetric with respect to the pixel grid in that direction and therefore this center coordinate coincides with the one of the fit. Nevertheless for all other cases we detect a tendency towards 0.5 as indicated by the arrows, which are amplified by a factor of 20. The fact, that the tendency goes towards 0.5 and not 0 or 1 is probably caused by starting the fitting iteration in the center of the local maximum pixel.

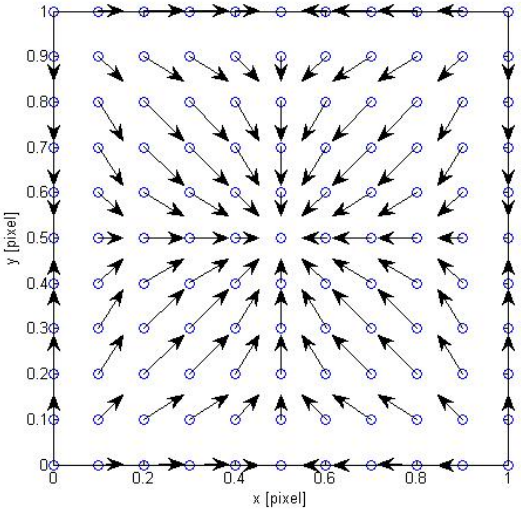

Figure 3.2: Shift of fitted spot centers due to pixelation, spot diameter 5 pixels.

For a spot size of 5 pixels (500 nm for the common pixel size of 100 nm) we measure an average systematic error of $4 \cdot 10^{-3}$ px (0.4 nm) and a maximal error of $6 \cdot 10^{-3}$ px (0.6 nm) already for fitting the perfect Gaussian. The existence of this systematic error coincides with the observed error difference between numerical integrations and pixelated algorithms (e.g. Figure 4.5). We will quantify the observed effects in Chapter 4.4.

All used algorithms were programmed in MATLAB®, for further information consider Appendix A.1.

4 Random STORM images

In this chapter we want to simulate STORM images, allowing us to know the exact spot centers to closely examine the fitting process. We still use MATLAB®, for the simulation code see Appendix A.1.

4.1 Image generation and setup

Again based upon Thompson [10], who states that the background noise can be considered as a Poisson process, one can use the following Algorithm 4.1.

Algorithm 4.1 Random image generation.

- Fix a spot center.
- Generate a fixed number of photons from a Gaussian distribution.
- Collect the photons on a coarse grid.
- Add Poisson distributed background photons for every pixel.

In theory one should use the Airy distribution for photon generation, however the difference between its central ring and a Gaussian distribution is very small, while we do not observe the outer rings in practice anyway. Parameters available for tuning are the spot center relative to the pixel grid, the number of photons in the spot, the pixel size, the size of the distribution and the average number of background photons per pixel.

4.2 Error dependencies

First of all we want to check how much the precision of the fitting is related to the position of the spot center within the pixel. This might depend crucially on the relation of the spot size, i.e. the diameter of the distribution and the pixel size as the sketch in Figure 4.1 shows.

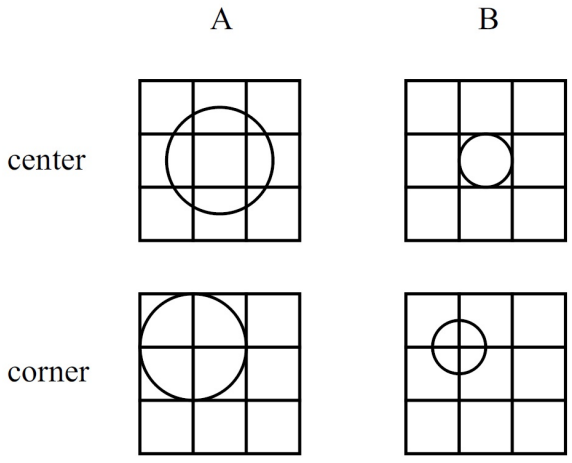

Figure 4.1: Dependency on the position of the spot center within the cell.

One may foresee that in case A, where the spot is twice as big as one pixel, better results are obtained for the pixel center. This is reasonable as photons hit three pixels in each direction instead of two for the center located in the corner and thus we have much more detailed information to fit. On the other hand in case B, where the spots are smaller than one pixel, it would be preferable to have the spots located close to the pixel corners to obtain photon hits in more than one pixel.

Therefore one has to consider realistic values for those sizes to be able to validate the influence of their relation. From (2.2) we know that the radius of the inner Airy disk is given by

$$r_{Airy} \approx \frac{0.61\lambda}{NA} \implies d_{Airy} \approx \frac{1.22\lambda}{NA},$$

where λ denotes the emission wavelength of the used fluorophore and NA the numerical aperture, a dimensionless characteristic of the objective. As observable in [13], Table 1, the emission wavelengths

of different fluorophores range from 500 to 800 nm. On the other hand numerical apertures of up to 1.51 are theoretically possible with oil immersions, while nowadays values between 1.0 and 1.35 are common in practice [3]. Hence the spot diameter may vary between 400 and up to 800 nm. Moreover light microscopes usually have pixel sizes of down to about 100 nm.

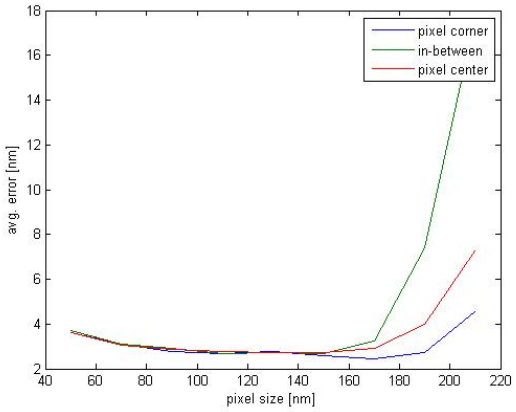

Figure 4.2: Average error according to pixel size; spot diameter 800 nm, 10000 photons, average of 130 photons/pixel background noise.

Thus in Figure 4.2 we plot the average error for positions close to the corner, in-between and close to the center of a pixel over the pixel size. For a constant spot size of 800 nm 1000 random images with respectively 10000 photons are generated, fitted with the Gaussian mask algorithm and the errors, here denoting the euclidean distance of the fitted center from the original one, averaged. A Poissonian background noise with a mean of 130 photons per pixel (as reported by our experimental contributors) is added. Keeping the spot size constant is reasonable as only the relation of spot size and pixel size is relevant. Additionally for a fixed fluorophore only the pixel size can be adapted.

Two effects can be observed here. On the one hand whenever the pixels are too small the spot is too far spread out and therefore gets hard to distinguish from the background noise. On the other hand for large pixels, depending on the position of the center within the pixel, the photons hit too few different pixels and become harder to recognize as suspected above. Nonetheless between these extreme cases there is a sufficiently large range of pixel sizes (60 - 140 nm), where small errors occur independent of the spot position.

The common pixel size of about 100 nm is right in the center of that interval and should therefore be adequate for the different occuring spot diameters. Anyhow we checked this fact, the result can be seen in Figure 4.3. As visible for small spots the position of the center becomes recognizable, still the error stays acceptable. Nevertheless it is remarkable that minization of the spot diameter does not imply the smallest possible average error. Apart from these effects at the lower limit, we perceive a linear increase of the error, which is independent of the center position.

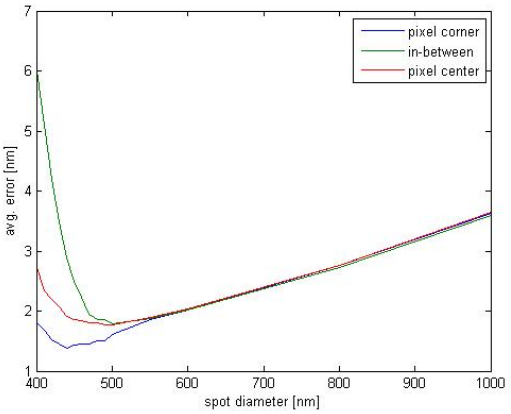

Figure 4.3: Average error according to spot diameter; pixel size 100 nm, 10000 photons, average of 130 photons/pixel background noise.

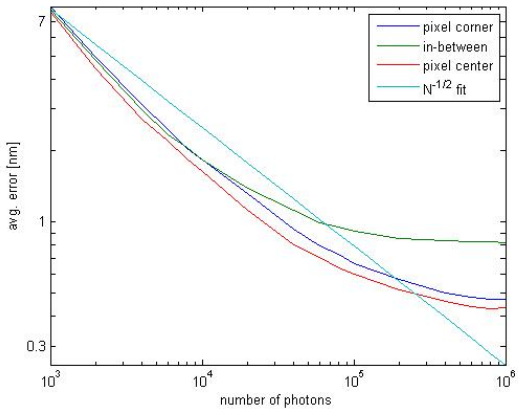

Figure 4.4: Average error according to photon number, pixel size 100 nm, spot diameter 500 nm, average of 130 photons/pixel background noise.

Finally we try to confirm the dependency of the average error on the number of photons in a spot in the form $\triangle x \sim \frac{1}{\sqrt{N}}$ for large enough N, which we introduced in Chapter 3.1, see (3.3). Our simulation results seem to agree with the estimated proportionality in the logarithmic plot in Figure 4.4 pretty well at first, however the decay is damped and thus the average error does not converge to zero.

4.3 Effect of numerical integrations

As we hope to avoid unnecessary approximations, we use random generated data with known spot centers to check whether our algorithm is more precise, i.e. yields a fit that is significantly closer to the true center. First, we fix the number of photons at 10000 and generate 1000 frames respectively for the occuring spot sizes. The pixel size is kept at 100 nm, the background noise is set to zero here as we want to compare with the minimal possible error, i.e. the error caused by coarse- and finiteness of the data, which is according to (3.3) given by

$$\langle \triangle x_{min} \rangle = \sqrt{\frac{\sigma^2 + \frac{a^2}{12}}{N}}.$$

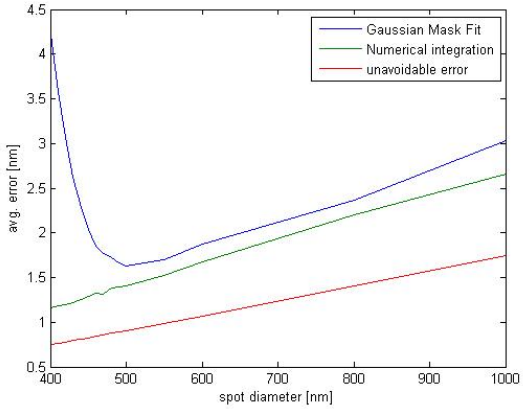

Figure 4.5: Average error according to spot diameter, pixel size 100 nm, 10000 photons, no background noise.

In Figure 4.5 we observe that at least 10% of the error can be avoided by using numerical integration fitting. Especially at small spot sizes the algorithm outperforms the Gaussian mask fit as the error keeps linearly decreasing here. This agrees with the theory, even if the error is still larger than the unavoidable one.

Additionally we verifiy that the average error caused by numerical integration fitting is independent of the position of the spot center within the pixel now. This can be seen in Figure 4.6. We add a Poisson distributed background noise with an average of 130 photons per pixel, all other parameters remain unchanged.

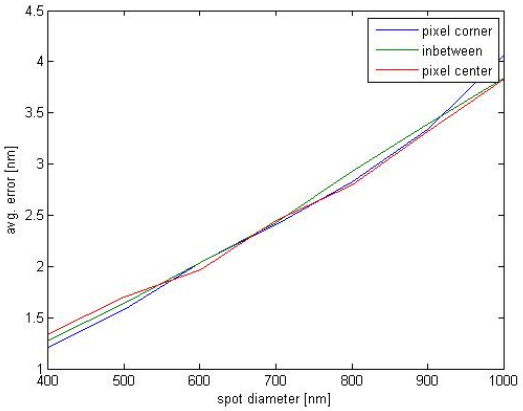

Figure 4.6: Average error according to spot diameter, pixel size 100 nm, 10000 photons, average of 130 photons/pixel background noise.

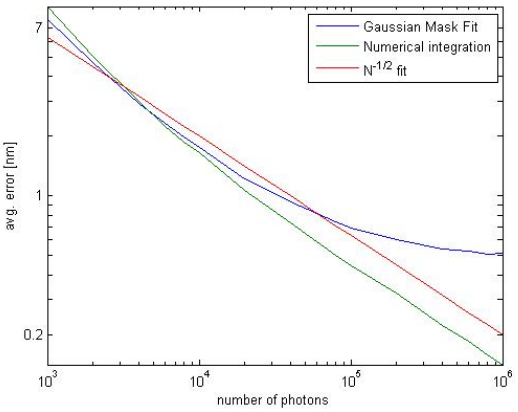

Figure 4.7: Average error according to photon number, pixel size 100 nm, spot diameter 500 nm, average of 130 photons/pixel background noise.

Finally we have a look at the relation of the average error and the photon number. We choose a constant spot size of 500 nm for which both algorithms are performing fine when using the common pixel size of 100 nm. Then we generate 2000 images for several photon numbers and add an average of 130 photons per pixel background noise. In Figure 4.7 we obtain decreasing errors for both algorithms, however the numerical integration fitting again performs better. Especially it fits the expected $N^{-1/2}$-dependancy very well, resulting in an average error converging straight to zero in contrast to the Gaussian mask fit.

4.4 Quantification of occuring shifts caused by pixelation

In Chapter 3.2 we analyzed the occuring systematic errors due to pixelation in the extreme case of fitting the perfect Gaussian distribution, which equals the limit of observing an infinite number of photons stemming from one spot. Now we want to fix different photon numbers and analyze the occuring shifts. Thus for every fixed photon number we generate 200 equally distributed spot centers and 200 STORM images respectively (40.000 STORM images altogether) and fit those. To increase the number of samples we consider x- and y-coordinate as independent, which is reasonable because a 2D Gaussian is only the combination of 1D Gaussians in each coordinate direction. Then we calculate the 1D shifts towards the spot center. Last but not least we need to be sure, which part of the error is caused by pixelation. Therefore we apply numerical integrations as a cross-check to the Gaussian mask fits.

The results for a spot size of 500 nm and a pixel size of 100 nm, for which both algorithms are performing fine, can be found in Figure 4.8 on the next page. No background noise is added here as it would only make our results less clear. We observe that the shifts for the NI algorithm are centered around 0, while the GM fits are unambiguously biased in direction of the spot center. As expected the distribution becomes less broad for an increasing photon number.

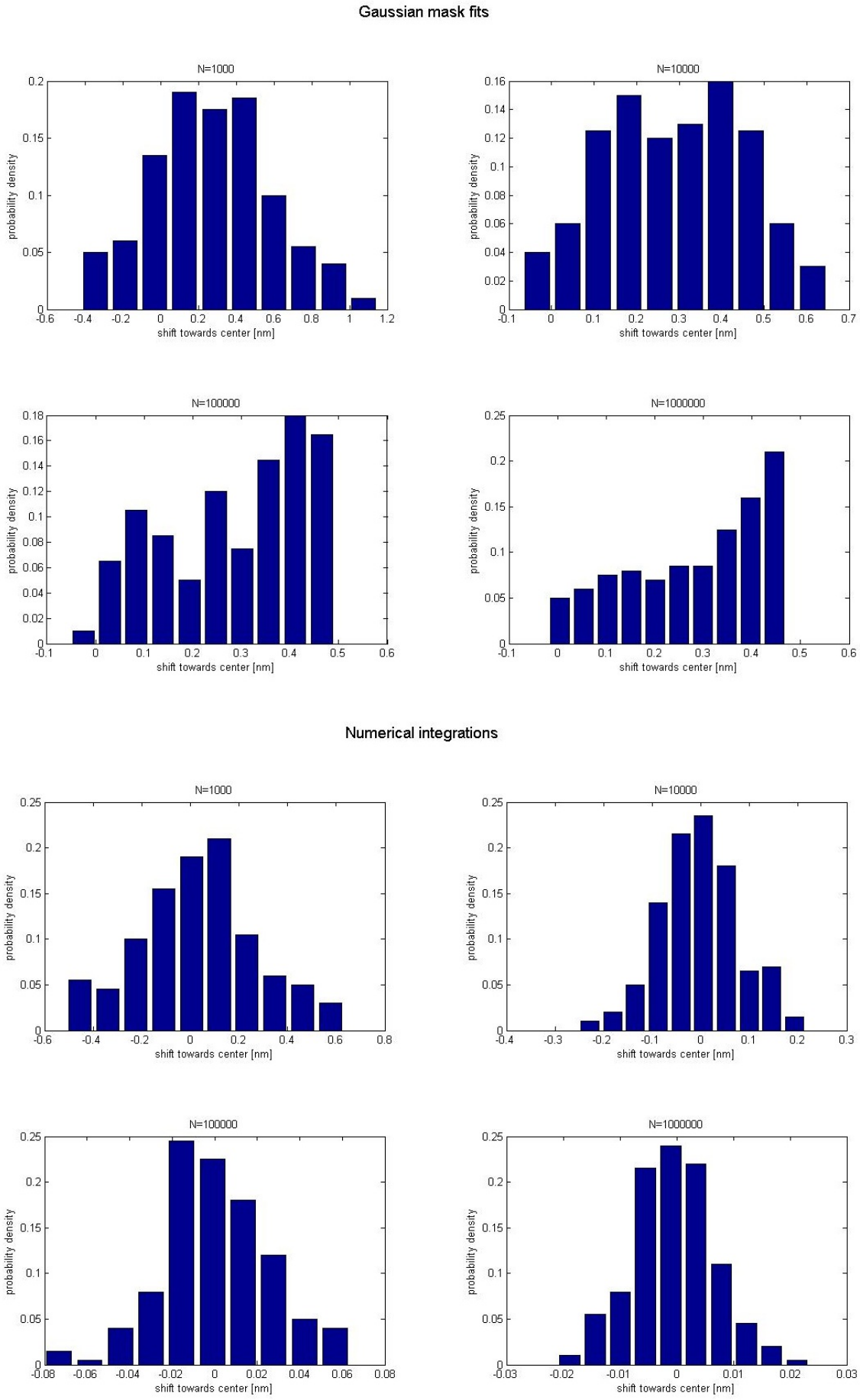

Figure 4.8: Distribution of 1D shifts, spot size 500 nm, pixel size 100 nm, no background noise.

Furthermore average and maximal shifts for all simulated setups are shown in Table 4.1. The displayed shifts are 1D, thus multiplication by $\sqrt{2}$ yields the particular 2D errors. We recognize that the average shift for the GM algorithm stays constant at approximately 0.28 nm (0.4 nm in 2D), which perfectly agrees with the value from Chapter 3.2. Anyhow the maximal error is steadily decreasing from 1.15 nm (1.63 nm) for 1000 photons down to 0.47 nm (0.66 nm) for 1000000 photons. This coincides with the 2D limit of 0.6 nm for the perfect distribution, too. On the other hand the maximal errors for the numerical integrations are strikingly smaller and converging to zero very fast.

number of	GMF		NI	
photons	mean [nm]	max [nm]	mean [nm]	max [nm]
1000	0,2797	1,1525	0,033	0,6341
10000	0,2871	0,6494	-0,0027	0,2158
100000	0,2867	0,4924	0	0,0638
1000000	0,2858	0,4704	-0,0005	0,0234

Table 4.1: 1D average and maximal shifts, spot size 500 nm, pixel size 100nm, no background noise.

These results show that the numerical integration algorithm is superior to pixelated approaches.

4.5 Fitting a Poissonian background

Back in our algorithm testing environment we want to compare the developed algorithm for Poissonian background fitting (cf. Chapter 2.4) with the former ones. Thus we use the same settings as in Chapter 4.3. We start off comparing different spot sizes for a fixed pixel size of 100 nm. 1000 frames with 10000 photons respectively are generated using no background noise. The unedifying results can be seen in Figure 4.9. As shown, there is no improvement compared to the common numerical integration fitting, which is simpler and faster.

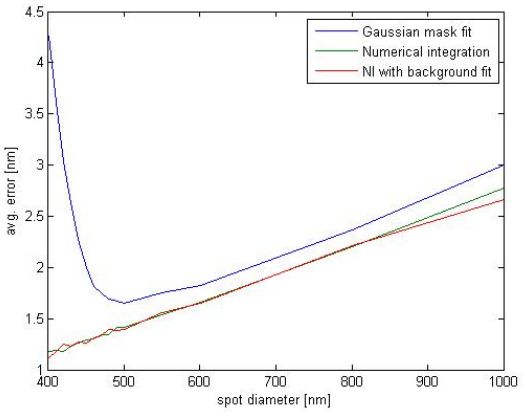

Figure 4.9: Average error according to spot diameter, pixel size 100 nm, 10000 photons, no background noise.

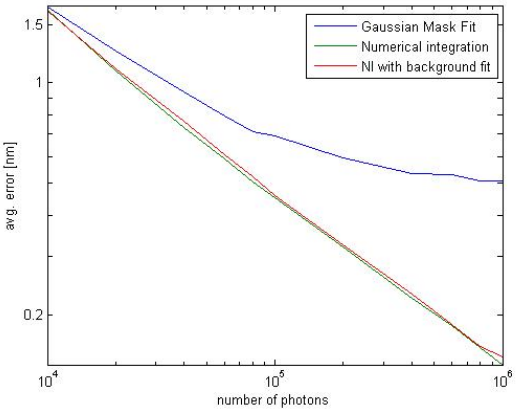

Figure 4.10: Average error according to photon number, pixel size 100 nm, spot diameter 500 nm, average of 130 photons/pixel background noise.

Next we analyze different photon numbers for a fixed spot size of 500 nm, using the common pixel size of 100 nm and generating 5000 frames respectively. The average background noise rate is set to 130 photons per pixel. Nonetheless Figure 4.10 shows no improvements either.

Overall we cannot see improvements in the localization precision due to Poissonian background fitting. The photon number can be fitted more accurately that way, yet in this thesis we are only looking for improved resolution as we deem the current photon number fits sufficient.

4.6 Simulation results for Gaussian background fitting

Finally we implement two variations of our latest algorithm:

- GbNi, our numerical integration algorithm with a Gaussian background fit starting in the center of the local maximum pixel and
- WpGbNi, using the fitting result of our common numerical integration (Ni) algorithm as an initial iterate for GbNi.

Now we generate 1000 STORM images for several photon numbers and fit them with the different algorithms. We add a Gaussian background noise with a mean of 130 and a standard deviation of 20 according to our noise statistics in Chapter 7.1. The spot size is set to 500 nm and the pixel size to 100 nm as usual. Unforunately we are not able to observe significant improvements in Figure 4.11. Thus we do not apply GbNi or WpGbNi to real data as the Ni algorithm seems to already achieve the same accuracy. This may be caused by the fact that variations of the local background values are much weaker than other noise factors such as pixelation and finiteness of the sample.

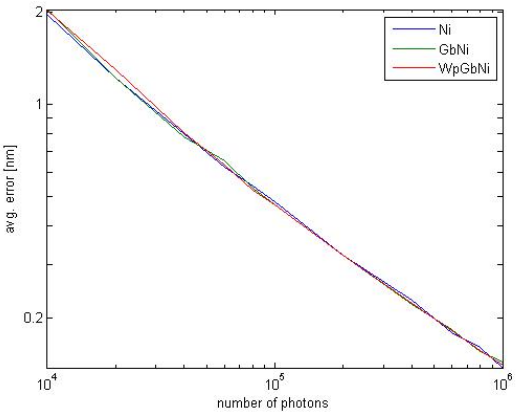

Figure 4.11: Average error according to photon number, pixel size 100 nm, spot diameter 500 nm, Gaussian background with mean 130 and standard deviation 20.

5 Processing experimental data

Having data of STORM runs at hand that were attained by Gregor Lichtner at the FMP Berlin, we first have to investigate how to handle experimental data.

5.1 Available fitting tools

Our collaborators currently use the software RapidSTORM, which was originally developed by Steve Wolter in the context of his diploma thesis in 2009 [14]. The package applies the so called Levenberg-Marquardt algorithm, which is a more robust alternative to the Gauss-Newton algorithm for solving least squares problems. Nonetheless, the distribution is fitted with a pixelated Gaussian as in Chapter 2.2. Although the algorithm might be faster or locate more spots, it does not possess a higher accuracy. The background noise fitting is done by subtracting a local mean value, which might be another opportunity for improvements.

Besides, there are several other frameworks for fitting STORM data. The two most famous alternatives are DAOSTORM [15] and QuickPALM [16], which date back to 2011 and 2010 respectively.

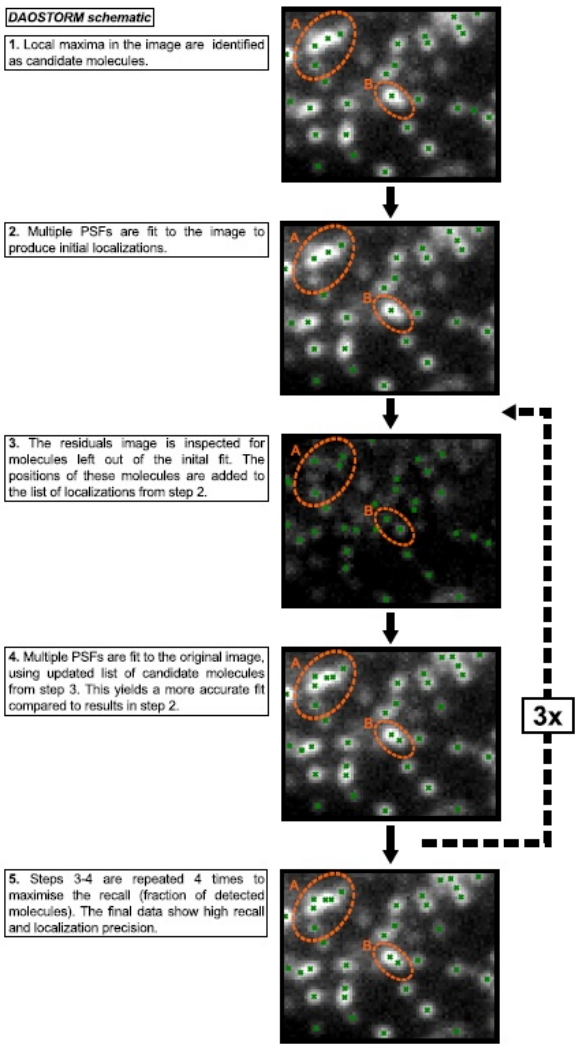

Figure 5.1: Schematic illustration of the DAOSTORM algorithm. [15]

The former is an adaption of an astronomy software, DAOPHOT II, which allows to fit overlapping molecules. This is accomplished by grouping up candidate spot centers with overlapping distributions and minimizing the total sum of the squared errors of all fits within a group. The employed PSF model relies on a pixelated Gaussian as well. Due to several identified noise sources, different ad-hoc weights are included in the least squares fitting. Albeit, the algorithm does not outperform the Gaussian mask estimation (cf. Chapter 3.1) for well-separated molecules in terms of precision, as stated in the article's supplement. The difference of DASTORM compared to the ordinary approach (cf. Figure 1.5) utilized in RapidSTORM and QuickPALM can be seen in Figure 5.1.

QuickPALM on the other hand is a plugin for the visualization software ImageJ, which allows real-time processing of STORM or PALM (PhotoActivated Localization Microscopy) data. In contrast, the precision is worse than for Gaussian fitting methods, since a modified center of mass algorithm is used to achieve this.

For localizing single molecules as precise as possible, RapidSTORM seems to be the best choice out of the currently available implementations. Furthermore it contains many interesting features for experimentalists, such as automatic rejection of "bad" fits.

5.2 Identifying trajectories

A so called STORM image usually consists of several (from 100 up to 10.000) frames displaying the same observed area. The imaging time for every frame is fixed, therefore the same spot is monitored in multiple consecutive images. Now one wants to detect those "trajectories". Thus we have given RapidSTORM output files, which contain a list of spot locations and photon numbers for every frame and look for reappearances. Combining those results in a more precise localization of the spot center as the photon number-weighted mean of the single locations. The detailed procedure is given by Algorithm 5.1.

Algorithm 5.1 Identification of trajectories.

- Get all localizations out of the first frame.
- Set these as starting points for new "active" trajectories.
- For all frames, repeat the following steps.
  - For all localizations within the frame:
    - Check whether there is an active trajetory within a fixed range of the localization.
    - If it is, add the localization to the trajectory.
    - If not, generate a new "active" trajectory starting in the localization.
  - Set all trajectories, that did not appear in the frame to "completed".
- Calculate the weighted localizations and total photon numbers for all trajectories.

5.3 Drift

Unforunately one cannot ensure that there is no drift of the observed sample, which may for example be caused by temperature changes or external forces. Even if the movements are very small in usual microscopic dimensions, they may become a non-negligible factor whenever we want to achieve nanometer resolution. Therefore Mlodzianoski et al. analyzed drifting effects in STORM for the 3D case in 2011 [17].

Analogously we want to estimate the sample movement for the 2D case now. Of course we do not estimate the shifts from localization to localization as then we would misinterpret the fitting inaccuracy as drift. Though grouping up of spots over multiple frames and analyzing the drift of the averaged spot centers should be an improvement for large enough frame numbers. Thus we will carefully pay attention to whether or not a data set is influenced by drift or not.

In case it becomes necessary, we will implement the following approach. First of all we use so called beads, "large" objects which can be tagged with numerous fluorophores, such that they are permanently emitting photons. Second we model the sample drift by a combination of translation and rotation. We denote the velocity with $\vec{v}$, the angular frequency with ω and the center of rotation with $\vec{r}$ and obtain the following evolution of a point $\vec{x_0}$ in time

$$\vec{x}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \vec{x_0} + \vec{v} \cdot t + \begin{pmatrix} \cos(\omega t) & -\sin(\omega t) \\ \sin(\omega t) & \cos(\omega t) \end{pmatrix} \cdot (\vec{x_0} - \vec{r}).$$

By locating at least four beads we could now calculate the seven free parameters of our modelled drift or whenever more of them are available perform a non-linear least squares fit. However in practice we are not interested in a continuous drift or knowing the center of rotation, but in an estimation of the drift from one frame to another. Therefore we can discard time as a factor and approximate the drift for a discrete step by

$$\vec{x_{n+1}} - \vec{x_n} = \vec{v} + M_{rot} \cdot \vec{x_n}. \tag{5.1}$$

Here M_{rot} is an arbitrary "rotation" matrix and as already mentioned $\overrightarrow{x_n}$ does not denote the localization of a spot in frame n, but a mean localization of the surrounding frames. Now every assigned pair $(\overrightarrow{x_n}, \overrightarrow{x_{n+1}})$ provides two equations to estimate the six parameters

$$\overrightarrow{v} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \text{ and } M_{rot} = \begin{pmatrix} m_1 & m_2 \\ m_3 & m_4 \end{pmatrix}.$$

of the drift $\overrightarrow{x_{n+1}} - \overrightarrow{x_n}$. Hence we need to locate at least three beads in the surrounding frames. In case of more localizations we calculate the linear least squares fit since the drift is no longer parameter-dependant in a non-linear way.

At this point we only have to describe how to obtain such a fit. Assume $i \in \mathbb{N}$ equations

$$a_{i,1}\lambda_1 + a_{i,2}\lambda_2 + ... + a_{i,k}\lambda_k = b_i$$

for known right hand sides b_i and factors $a_{i,k}$ are given and we want to estimate the $k \in \mathbb{N}$ unknowns $\lambda_k \in \mathbb{R}$. This can be accomplished by minimizing the error in the Euclidean norm, i.e.

$$||b - A\lambda||_2 \leq ||b - Av||_2 \ \forall v \in \mathbb{R}^k.$$

Now according to [18], Theorem 2.14, $\lambda \in \mathbb{R}^k$ is a minimizer, if and only if

$$A^T A\lambda = A^T b \tag{5.2}$$

Furthermore it is unique if A is injective.

In our case the right hand sides are the occuring drifts, the unknowns are the free parameters and the respective factors follow from (5.1):

$$\begin{pmatrix} b_{2i} \\ b_{2i+1} \end{pmatrix} := \begin{pmatrix} x_{n+1,i} \\ y_{n+1,i} \end{pmatrix} - \begin{pmatrix} x_{n,i} \\ y_{n,i} \end{pmatrix},$$

$$\lambda := (v_1, v_2, m_1, m_2, m_3, m_4),$$

$$a_{2i} := (1, 0, x_{n,i}, y_{n,i}, 0, 0) \text{ and}$$

$$a_{2i+1} := (0, 1, 0, 0, x_{n,i}, y_{n,i}).$$

As we will not locate two spots in the exact same place A is obviously injective and thus a unique minimizer is given by (5.2), which can be evaluated using basic linear algebra.

6 xStorm

In the context of this thesis an independent software package for spot localization was created.

6.1 Necessity of the development

Even if RapidSTORM is open source code and the possibility of including other "fitting kernels" was originally build in, there are several reasons against that. Above all, the code was once well-structured, but soon became confusing due to several bug- and hotfixes and only sparse annotations. Furthermore the documentation is mainly meant for users, while the code contains many parts that are not relevant for the present thesis.

Consequently we decided to develop our own fitting environment for experimental data called xStorm, where x is an abbreviation of exact. In particular this allows us to include several algorithms for the analysis of external influences such as noise or drift. Nonetheless Steve Wolter's diploma thesis [14] and the article published by his group on their experiences and results while designing and working with RapidSTORM [19] were very helpful.

In terms of the programming language we decided in favor of C++, which combines computational efficiency with availability of diverse functions, extensions and packages, e.g. for user interfaces and multithreading. In addition the G++ compiler guarantees the portability of the C++ code to the most common operating systems. Fortran would have been a reasonable decision as well, however making computational power accessible for the user as well as programming itself is more complex here. MATLAB®, which we used for our simulations, is very intuitive as it has many mathematical functions already built-in, but it is simply too ineffecient (i.e. slow) when it comes to working with huge amounts of data. Similarly, these computational disadvantages apply to Java and Python.

6.2 Input and output

As an output from a STORM experiment, one obtains a .tif- or .tiff-file containing several frames. We will have a closer look at this format in the following Chapter 6.3, for now we just assume that we can extract framewise matrices of intensity values from it. Furthermore we need to know the pixel size in nanometers to provide meaningful final results. Optionally we would like to know the emission wavelength (λ [nm]) and the numerical aperture (NA) of the used microscope, because this gives us the spot size (cf. (1.1)).

One fit finally consists of only four numbers. Those are x- and y-coordinates of the spot center, the number of fitted photons and the frame the spot was found in. Consequently we use a simple text file with four columns (and a space as separator) to print our results and condense a complete STORM image into a list of found spots, alike the output format of RapidSTORM.

6.3 The tagged image file format (TIFF)

Nowadays the TIFF™ is controlled by Adobe Systems and known as a flexible file format that uses so called header tags to structure data and multiple images. The latest version (6.0) of its specification [20] can be found on Adobe's homepage.

Unsurprisingly the first task when developing a framework for STORM experiments is being able to extract data from these images, such that one can efficiently work with them. Therefore understanding the TIFF-structure displayed in Figure 6.1 is crucial.

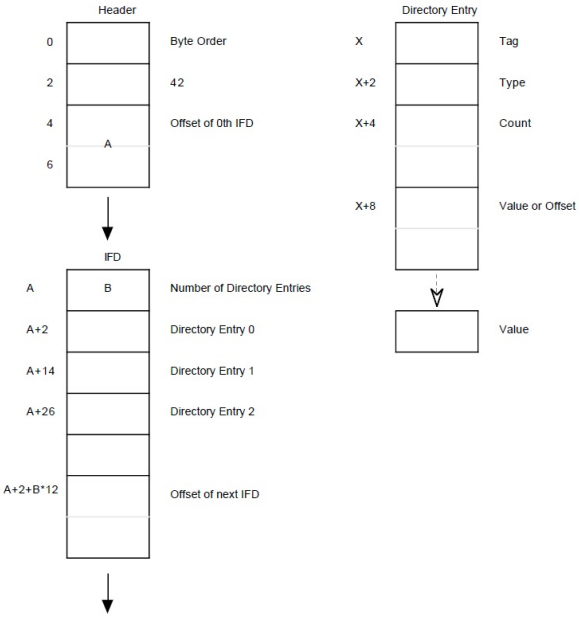

Figure 6.1: The TIFF structure. [20]

The first eight bytes contain two characters defining the byte order (little or big endian), two containing the number 42 to label the file as a TIFF and four with the offset (distance to the beginning of the file) of the first Image File Directory or shortened IFD. In turn each IFD starts with two bytes containing the number of 12-byte directory entries following and ends with four bytes describing the offset of the next IFD. The last IFD ends with four bytes of zero. Finally each directory entry consists of the tag identifying the field (two bytes), the field data type (two bytes), the number of values (four bytes) and the value(s) itself or if those exceed four bytes a file offset to where the values can be found.

An example directory entry in big endian is

$$\underbrace{1\ 0}_{tag}\ \underbrace{0\ 4}_{type}\ \underbrace{0\ 0\ 0\ 1}_{\#values}\ \underbrace{0\ 0\ 1\ 224}_{values/offset}\ .$$

Having in mind that each byte may contain the values 0..255, we detect the field as tag number 256, which (according to the specification) contains the number of pixels in x-direction. The type 4 stands for LONG, i.e. 4-byte unsigned integer and the number of values is 1. Thus $1 \cdot 4 = 4$ bytes are necessary for the contained value(s) and the last four bytes contain the value itself, which is $1 \cdot 256 + 224 = 480$. As a result the described image has a width of 480 pixels.

Apart from these tags only raw data is contained in the file, while usually each IFD specifies where the data for one frame can be found and how it is formatted.

6.4 Program design

For the implementation the programming framework Ultimate++ was chosen. It is available for several platforms as it is compatible with G++ and contains all libraries we need as well as its own development environment, TheIDE.

The basic code (cf. Appendix A.2) was split up according to the following tasks.

- Graphic User Interface (GUI)- xStorm.h, xStorm.lay
- TIFF processing - tiffproc.h
- Self-written datatypes - data.h
- Routines for working with a single frame, i.e. spot finding and fitting - spots.h

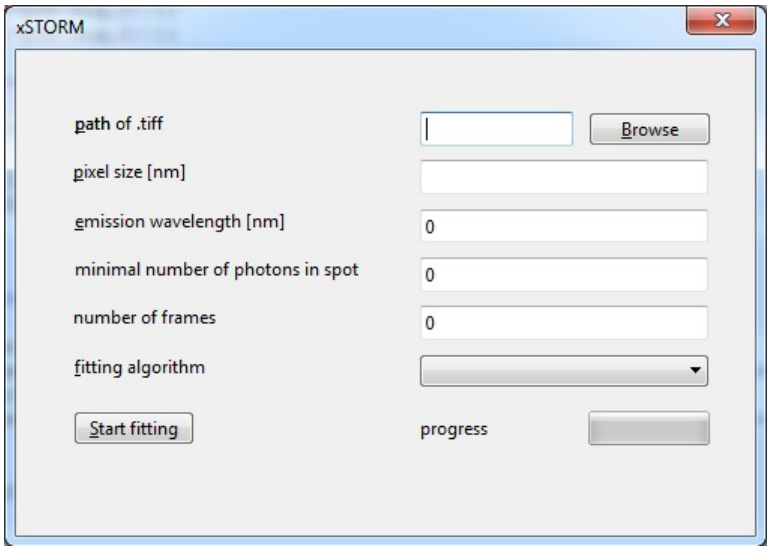

Figure 6.2: The xStorm GUI.

As we are interested in comparing different fitting algorithms the program allows selection of one of them. Furthermore different fitting parameters need to be specified as can be seen in the GUI in Figure 6.2. Those are

- the pixel size in nanometers,
- the emission wavelength in nanometers, which specifies the approximate spot size; 0 means that the spot size (or equivalently the standard deviation of the Gaussian) is included in the fit,
- the minimal number of photons; 0 means that the program fits all spots it can recognize and
- the number of frames to be fit; 0 means that all contained images are processed.

6.5 Parallel processing

The fitting procedure contains several computationally expensive steps. Therefore multithreading should be implemented wherever reasonably possible. Fortunately the fitting process allows a high level of parallelization, resulting in an appreciable decrease of computing time already on a quad-core CPU. The multithreading is done using the "CoWork" class contained in Ultimate++.

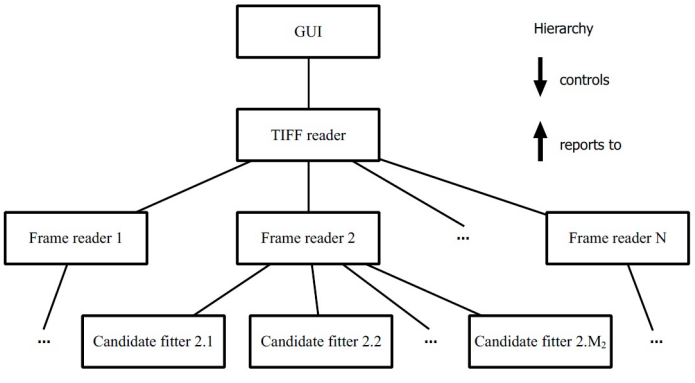

Figure 6.3: Multithreading in xStorm.

Of course only one thread can access the source file at the same time. Still whenever a complete frame has been read, it can be passed to a new thread to deal with it, while the original one continues reading. The new threads scan the frames for spot candidates. Ultimately, for every detection a surrounding region depending on the spot size, called subframe, can be cut out and transferred to a new "subthread".

Additionally the reader thread is connected to a GUI thread, which keeps the GUI accessible while the fitting is done. A schematic overview is given in the adjacent Figure 6.3.

7 Experimental observations

In the end we analyze the experimental data. Some basic results are displayed here.

7.1 Noise statistics

As a start we want to have a look at the background noise. To do so we consider all pixels within a 40×41 grid of a common STORM image (#1, 100 frames), that were part of no fitted spot. Then we collect the occuring intensities, which are proportional to the photon numbers, into several bins. Finally normalizing the data results in the attached Figure 7.1.

The common statistical formulas yield a mean value of $\lambda = 578$ and a standard deviation of $\sigma = 81$, which correspond to the plotted graph. In contrast to Thompson's assumption [10] this is no Poisson distribution at all, as $\sigma^2 = 6561 \gg 578 = \lambda$. However the Gaussian fit seems accurate. It remains questionable why none of the authors of the following literature examined that fact, even if a few (e.g. [19]) use non-Poissonian background.

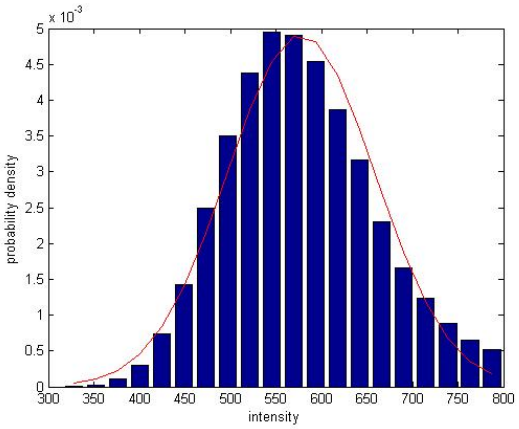

Figure 7.1: Experimental background noise fitted with a Gaussian.

The small imbalance of the distribution to the right, which results in a minor shift of the Gaussian fit, is easily explainable as there are of course some pixels which were hit by spot photons though not treated as part of a spot. This applies especially to pixels beeing located in one of the outer rings of an Airy distribution.

One has to pay attention to the difference between the intensity and the number of photons here. Assuming a mean of 130 photons per pixel, the proportionality yields a standard deviation of approximately 20.

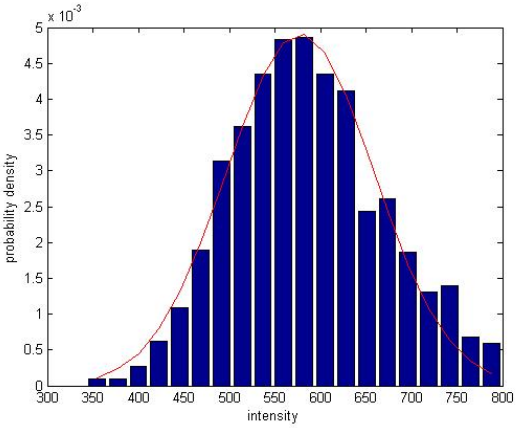

Figure 7.2: Background noise for a single frame compared to the overall fit.

To ensure that the increased standard deviation is not the result of a changing mean value of a Poissonian background distribution, we check the distribution of the background noise for a single frame. We arbitrarily choose frame 50 - no special effects should occur here. As Figure 7.2 depicts it agrees with the Gaussian fit of the whole background (red curve) very well, in fact for the single frame we obtain a mean of $\lambda = 585$ and a standard deviation of $\sigma = 83$ only slightly differing from the average ones.

Thus the background distribution is clearly Gaussian, but not Poissonian. As a consequence we discard our Poissonian background fit (cf. Chapter 2.4) from further research.

7.2 Single bead

Second we observe and localize a large single bead in STORM image #1 for 100 frames, being interested in the fit accuracy of the two already compared algorithms, numerical integrations(NI) and Gaussian mask fitting(GM).

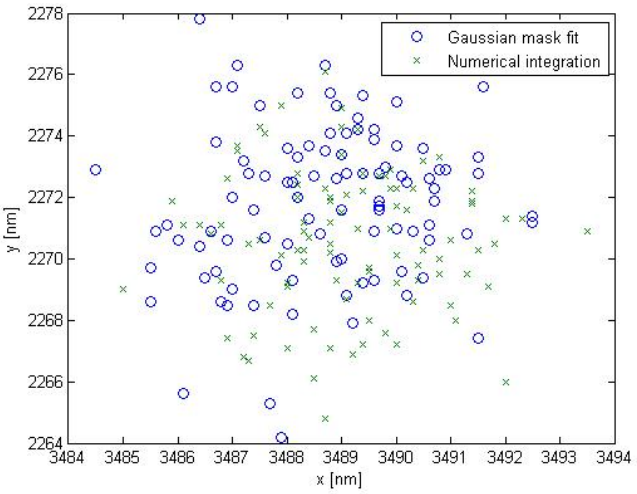

Figure 7.3: Distribution of fitted centers of a single bead.

The adjacent Figure 7.3 shows the distribution of the found spot centers for both algorithms. Calculations yield $\sigma_{NI} = 2.76$ and $\sigma_{GM} = 3.02$ as standard deviations of the numerical integration and Gaussian mask fits from their mean center respectively. Repeating the analysis using RapidSTORM results in $\sigma_{RS} = 2.89$. This agrees with our expectations because full least squares fitting (of a pixelated Gaussian) is applied here. Therefore our algorithm seems to perform best, primarily when we take into consideration that the average center of the numerical integration fit should be closer to the true spot center according to our simulations. This is a result we would like to quantify and confirm in theory.

Prior to this, we want to have a look at the occuring drift. Thence we average twenty fits respectively and plot the "movement" of those mean spot centers in Figure 7.4. We clearly see that for both algorithms a drift of approximately 1.5 nm in x- and 3 nm in y-direction occurs. Therefore we correct the influence of this drift by shifting every fit according to the average position of the surrounding 20 frames.

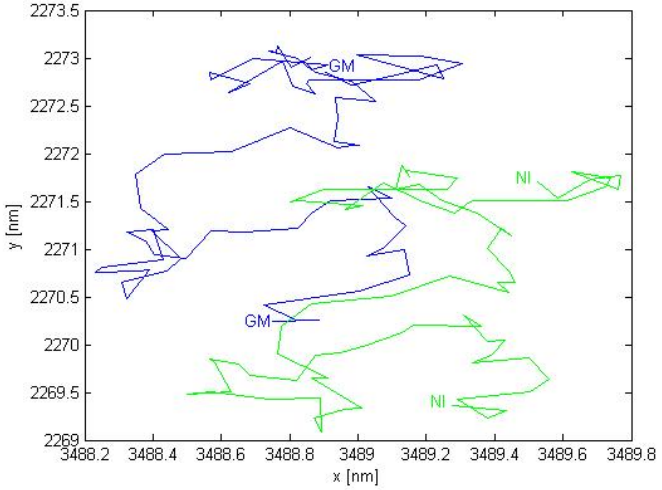

Figure 7.4: Drift of a single bead.

Figure 7.5 shows the drift-corrected spot centers in comparison to the previous Figure 7.3. We obtain considerably improved standard deviations of $\sigma_{NI} = 2.51$ and $\sigma_{GM} = 2.75$. To be able to relate this to the results of our simulations, we calculate the average errors ($\mathbb{R}^2$ distances from the mean center), too. A comparison is shown in the adjacent Table 7.1.

average error [nm]

	GM	NI
experimental data	2,67	2,47
drift-corrected data	2,43	2,23
simulation data	2,24	1,94

Table 7.1: Average errors for fitting a single bead.

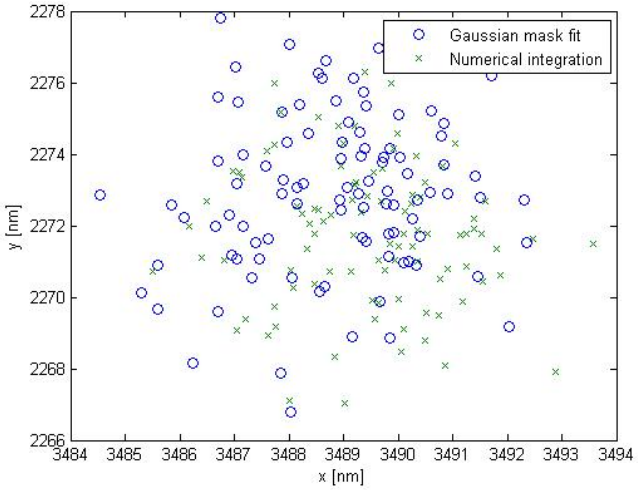

Figure 7.5: Distribution of fitted centers of a single bead after drift correction.

We estimate the spot diameter by 10 pixels (enlarged by the bead itself), know that the pixel size of our microscope is 105 nm and detect a mean intensity of 149400 per frame. Furthermore we know the underlying background distribution from the previous Chapter 7.1. Plugging this into our simulation environment we obtain average errors, which agree with the experimental results very well. We assume that the remaining difference results from the uncertainty caused by the drift correction and the difference between Airy disk (real) and Gaussian distribution (model).

7.3 Multiple beads

To quantify our the findings from the previous Chapter 7.2 we want to use several smaller beads (resulting spot size of approximately 8 pixels), fit them with both xStorm algorithms and compare the fitted intensities and the standard deviations of the fitted centers from their algorithm's mean. The results for the 100 frames of STORM image #2 are shown in the following Table 7.2.

bead number	average intensity per frame			standard deviation [nm]		
	NI	GMF	NI-GMF	NI	GMF	NI-GMF
1	17550	16762	787	7,38	7,50	-0,12
2	18695	17902	793	7,06	7,30	-0,23
3	84220	80189	4031	6,02	5,97	0,05
4	17733	16959	774	7,32	7,86	-0,54
5	22453	21246	1208	7,59	7,71	-0,12
6	37209	35565	1644	6,21	6,59	-0,38
7	84258	80416	3843	5,64	5,96	-0,32
8	43348	41397	1950	6,26	6,41	-0,16
9	48639	46442	2198	6,43	6,79	-0,36
10	46531	44326	2205	6,28	6,70	-0,42

Table 7.2: Quantitative comparison of different algorithms for bead fitting.

For all ten beads the NI algorithm is able to assign considerably more photons to the recognized spots, underlining the fact that its average spot center is most probably closer to the true one. This agrees with our simlutations. In addition our algorithm is able to obtain a smaller standard deviation for nine out of ten beads and fails only slightly for bead three.

On the other hand the obtained standard deviations are considerably larger than our simulations predict. For example, the generation of 100 images with an intensity of 40000 per frame but otherwise analogous setting yields the values displayed in the following Table 7.3 compared to weighted standard deviations for beads six, eight, nine and ten. Thus we want to survey the STORM image for drift effects.

standard deviation [nm]

	GM	NI
experimental data	6,63	6,30
drift-corrected data	6,65	6,29
simulation data	4,10	3,74

Table 7.3: Standard deviations for fitting multiple beads.

For this purpose we proceed as before by averaging twenty fits respectively and monitoring the "movement" of the mean spot centers of one bead. The disappointing result is shown in Figure 7.6, which cleary reveals that the spot seems to oscillate for more than 20 nm. Even taking more (checked for up to 50) frames for the averaged centers does not allow us to detect a drift in the original sense. Unfortunately this observation is independent of the applied fitting algorithm and the chosen bead, while the oscillations of the individual beads do not coincide (cf. Figure 7.7).

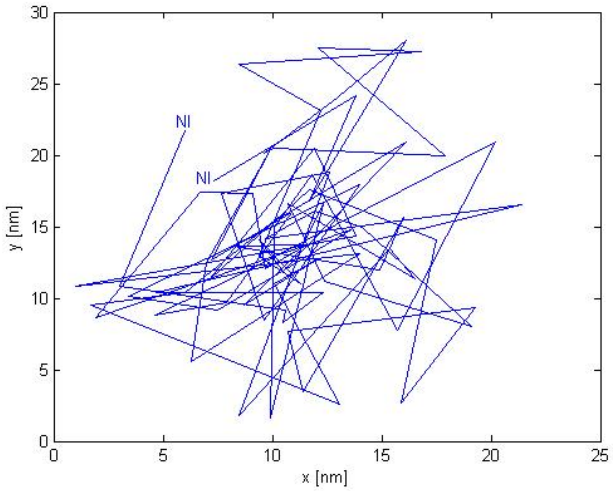

Figure 7.6: Movement of one of the beads.

Consequently we cannot achieve agreement of the experimental results with our simulations here, though we interpret the insufficient imaging quality as responsible. This is reasonable as satisfactory results (i.e. beads that do not move apart from the drift of the whole sample) are available (cf. the previous Chapter 7.2).

Nonetheless the numerical integration algorithm performs better than the Gaussian mask fit in terms of fitting accuracy and assigned intensity.

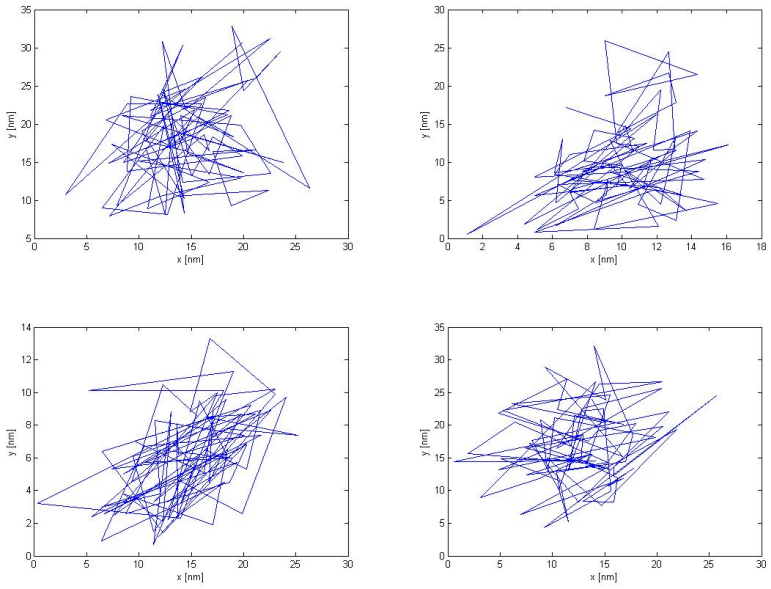

Figure 7.7: Movement of four of the beads.

7.4 Antibody fitting benchmark

Last but not least our experimental collaborators provided us with a STORM image (#3, 100 images) of many antibodies (and several beads for drift correction) as a benchmark test for the resolution. In the given sample the antibodies were heavily diluted to make them clearly distinguishable. A subset is manually selected and all spots in a surrounding region fitted and assigned. Then the standard deviations of the localizations for all selected antibodies are calculated. Finally the intensity-weighted average of these values determines the resolution of the applied fitting algorithm.

We obtain the average standard deviations of $\sigma_{NI} = 12.50$ and $\sigma_{GM} = 13.27$ for the fitting results of the respective algorithms using xStorm and $\sigma_{RS} = 14.29$ for the corresponding fits of rapidSTORM. It is not clear why rapidSTORM performs even worse than our Gaussian mask implementation here, but the problem is probably related to the spot finding algorithm. Indeed rapidSTORM identifies exceptionally many spots close to the image border, which our algorithms classify as artifacts. However we only use antibodies recognized by both programs to allow a fair comparison.

standard deviation [nm]

	GM	NI	RS
experimental data	13,27	12,50	14,29
drift-corrected data	11,75	11,12	12,62
simulation data	9,39	8,73	-

Table 7.4: Standard deviations for fitting antibodies.

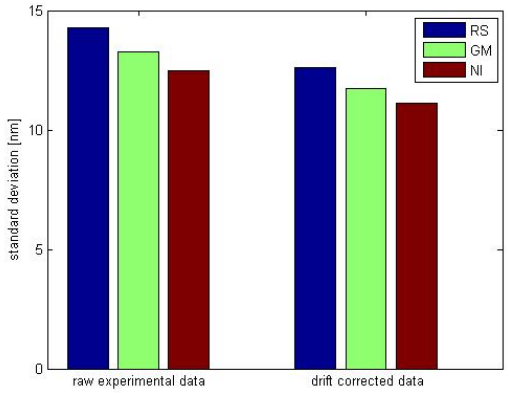

Figure 7.8: Antibody fitting benchmark results for different fitting algorithms.

Once again we compare the obtained values to the corresponding simulation results in Table 7.4, though the RapidSTORM fitting algorithm is not available in our MATLAB® environment. We expected smaller standard deviations, even though we did not model the drift and use a Gaussian instead of an Airy disk for image generation - however the values can still be considered as "consistent".

In addition a clean drift as in Chapter 7.2 can be recognized, resulting in an improvement of the standard deviation by about 1.5 nm independent of the applied fitting algorithm as visualized in Figure 7.8.

As a result our algorithm turns out to be the best choice for practical applications as well.

7.5 Bias due to the fitting method

Additionally we want to check the distribution of the fitted spot centers for STORM experiments with respect to the pixel grid. In theory we should observe a uniform distribution for a sufficiently large number of spots. Thus we fit all detectable spots within STORM image #2 (100 frames), altogether about 2500 spot centers. To increase the number of samples we then only consider a 1D distribution of a subpixel-coordinate around the pixel center. This allows us to use each fitted x- and y- coordinate as a sample respectively. As we want our results to be comparable we performed Gaussian mask fits (GM) and numerical integrations (NI) for the same spots.

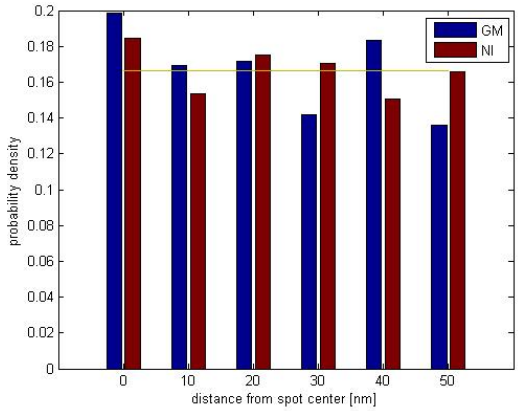

Figure 7.9: Distribution of fitted spot centers in experimental data.

The result can be seen in Figure 7.9, the histogram was normalized such that the observed distributions should be approximately uniform. Even if it is not obvious, the expected shift towards the spot center for the GM algorithm is identifiable, mainly in the large difference of fits close to the center compared to those in the pixel corners. Our NI fits on the other hand come considerably closer to a uniform distribution as χ^2 (the sum of squared errors) for our algorithm is at $8.14 \cdot 10^{-4}$ compared to $2.86 \cdot 10^{-3}$ for the GM fits.

Finally we want to test RapidSTORM for the assumed tendency towards the pixel borders. We consider x- and y-coordinate separately here as there may be different effects depending on the coordinate direction. To gain a larger number of samples we use another STORM image (#4, 10000 frames) and fit all (approximately one million) detectable spots. Then we repeat the above analysis to obtain Figure 7.10. As we see there is a clear tendendy towards the pixel borders, especially in the x-coordinate.

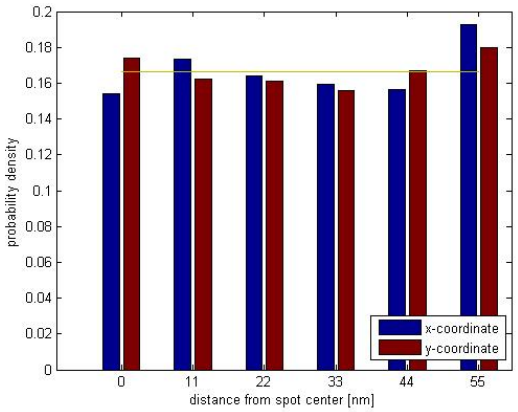

Figure 7.10: Distributions of fitted spot centers in RapidSTORM fits.

Consequently we have to expect shifts for fitting experimental data with pixelated approaches as predicted - another factor underlining the superiority of our numerical integration algorithm.

8 Summary and Outlook

Finally we want to briefly review our results and summarize them. Additionally we discuss possible further improvements and other approaches.

8.1 Our results

The numerical integration algorithm established in Chapter 2.3 is able to outperform pixelated approaches, such as the Gaussian mask fit(cf. Chapter 2.2) in terms of fitting accuracy as showed in Sections 4 and 7. Regrettably the improvements are not as significant as we had hoped. Nonetheless we avoid a systematic error, which underlines the remarkability of the slightly improved standard deviations and average errors. Furthermore our algorithm is stable for small spots (compared to the pixel size) in contrast to pixelated approaches, which cause non-negligible errors depending on the subpixel position of the spot center as observable in Figures 4.4 and 4.7.

Then we discovered that the background noise is not Poisson distributed as assumed, but can be adequately fitted with a Gaussian. This contradicts previous publications, though we are not able to employ this to clearly improve the fitting accuracy. Using the average background value as a fit seems imprecise, however we assume that the variations caused by the background noise are too small compared to other influences.

Moreover we detected drifts within a STORM image and increased the fitting accuracy by correcting them. A similar technique was already applied by other authors in 3D, still we were able to reconstruct their results in 2D.

Finally the mentioned existence of systematic errors induced by pixelated model functions was shown in Chapter 3.2 and the occuring errors quantified. Such an analysis was not carried out so far as our numerical integration algorithm is the first method avoiding pixelation.

The joint significance of these new findings motivates our intention to publish them.

8.2 Future possibilities

From our point of view the numerical integration algorithm is achieving the best currently feasible fitting accuracy for iterative approaches that try to directly fit a PSF to the observations. All known noise sources have been examined and their influence compensated as far as possible. Indeed the algorithm shows no bias and attention was paid to the background noise and drift effects. Nevertheless the errors caused by coarse- and finiteness of the sample are unavoidable for such algorithms.

On the other hand there may be sources of inaccuracy that we simply did not model. Thus improvements based on new physical research cannot be ruled out.

Apart from these efforts, Larkin and Cook recently published another completely different method for fitting STORM images [21]. Their basic idea depicted in Figure 8.1 is to assign a probability distribution to every photon (which describes where its source spot could have been) and join these distributions to describe the original spot center. This technique is faster than iterative algorithms and as they claim more precise for noisy images. For high signal-to-noise ratios however it perfoms worse than a maximum likelihood method. As such ratios can be attained in experimental setups, we do not interpret the algorithm as superior. Anyhow the approach may prove to be a signifcant contribution to superresolution microscopy, especially if it is extendable.

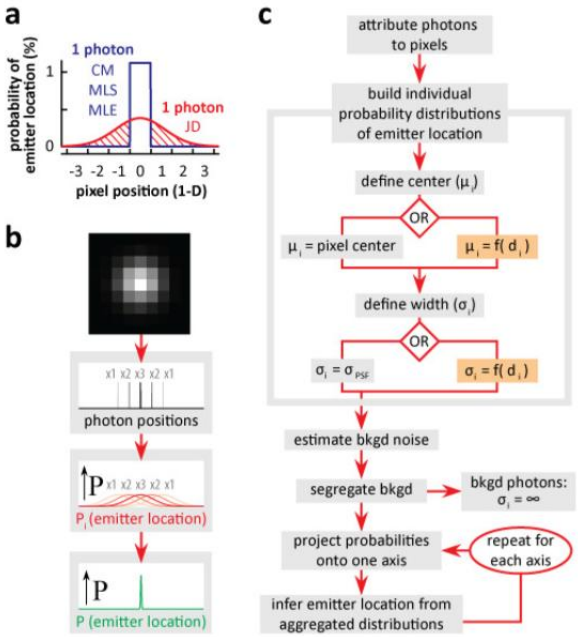

Figure 8.1: An alternative approach for fitting STORM images. [21]

Altogether, we see that the steady progress of improving microscopy resolution has not come to its end and hope that this thesis will be a small contribution.

A Code section

A.1 Simulation environment

All the simulation environment code is written in MATLAB®. For ecological reasons we decided not to print code in here, however the algorithms and the basic scripts for analysis and comparison will be archived by the Computational Molecular Biology (CMB) group at FU Berlin. In case you are interested in working with the algorithms or codes, feel free to contact schaller.cf@googlemail.com, but please keep in mind that it is research code, which may throw exceptions or provide senseless results caused by wrong inputs.

As we are still interested in the topic, you may ask questions concerning fitting algorithms as well.

A.2 xStorm

The same argument holds for the xStorm code. The source code and a running version will be archived, however no public release is planned. Thus the program might be very sensitive to several parameters or show unstable behaviour for cases which were not considered so far. Nevertheless if you are interested in working with xStorm or the algorithms in C++, contact us via mail and we will provide you with further information.

To compile xStorm we used Ultimate++ in combination with the G++ compiler contained in the GCC package.

B Assigning the matching background values

We briefly proof that the χ^2-sum is minimized by assigning the background values to the differences ordered by size. This is easily seen by the repeated application of the following formula.

Claim: $m_1 > m_2 \wedge l_1 > l_2 \Longrightarrow (m_1 - l_1)^2 + (m_2 - l_2)^2 < (m_1 - l_2)^2 + (m_2 - l_1)^2$

Proof: $(m_1 - l_1)^2 + (m_2 - l_2)^2 < (m_1 - l_2)^2 + (m_2 - l_1)^2$

$\Longleftrightarrow m_1^2 - 2m_1l_1 + l_1^2 + m_2^2 - 2m_2l_2 + l_2^2 < m_1^2 - 2m_1l_2 + l_2^2 + m_2^2 - 2m_2l_1 + l_1^2$

$\Longleftrightarrow m_1l_1 + m_2l_2 - m_1l_2 - m_2l_1 > 0$

$\Longleftrightarrow \underbrace{(m_1 - m_2)}_{>0} \cdot \underbrace{(l_1 - l_2)}_{>0} > 0$ □

Acknowledgments

First and foremost I want to thank my supervisor Prof. Dr. Frank Noé for the opportunity to research this compelling topic. In addition I am grateful for his and Gregor Lichtner's support throughout the last year. They were always amenable to my questions, eager to answer them and animated me to follow my own ideas.

Further thanks go to my proofreaders, particularly Joscha Podlesny, who assisted me in revising this thesis linguisticly and all other people who made my studies a great, enjoyable time.

Last but not least, I appreciate my parents' diligent encouragement over the course of my studies.

References

[1] "Airy disk - wikipedia, the free encyclopedia." `http://en.wikipedia.org/wiki/Airy_disk`. Accessed June 10, 2013.

[2] G. B. Airy, "On the diffraction of an object-glass with circular aperture," *Transactions of the Cambridge Philosophical Society* **5**, pp. 283–291, 1835.

[3] M. W. Davidson, "Numerical aperture." `http://www.microscopyu.com/articles/formulas/formulasna.html`. Accessed June 27, 2013.

[4] D. Thomann, D. R. Rines, P. K. Sorger, and G. Danuser, "Automatic fluorescent tag detection in 3D with super-resolution: application to the analysis of chromosome movement," *Journal of Microscopy* **208**(1), pp. 49–64, 2002.

[5] B. Zhang, J. Zerubia, and J. Olivo-Marin, "A study of gaussian approximations of fluorescence microscopy PSF models," *Proceedings of the SPIE* **6090**, pp. 104–114, 2006.

[6] B. Huang, H. Babcock, and X. Zhuang, "Breaking the diffraction barrier: super-resolution imaging of cells," *Cell* **143**(7), pp. 1047–1058, 2010.

[7] M. J. Rust, M. Bates, and X. Zhuang, "Stochastic optical reconstruction microscopy (STORM) provides sub-diffraction-limit image resolution," *Nature Methods* **3**(10), pp. 793–795, 2006.

[8] T. J. Gould and T. Hess, S, "Nanoscale biological fluorescence imaging: Breaking the diffraction barrier," *Methods in Cell Biology* **89**, pp. 329–358, 2008.

[9] M. Bates, B. Huang, G. T. Dempsey, and X. Zhuang, "Multicolor super-resolution imaging with photo-switchable fluorescent probes," *Science* **317**(5845), pp. 1749–1753, 2007.

[10] R. E. Thompson, D. R. Larson, and W. W. Webb, "Precise nanometer localization analysis for individual fluorescent probes," *Biophysical journal* **82**(5), pp. 2775–2783, 2002.

[11] M. K. Cheezum, W. F. Walker, and W. H. Guilford, "Quantitative comparison of algorithms for tracking single fluorescent particles.," *Biophysical journal* **81**(4), pp. 2378–2388, 2001.

[12] W. J. Cody, "Rational chebyshev approximations for the error function," *Mathematics of Computation* **23**(107), pp. 631–637, 1969.

[13] J. S. Silfies, S. A. Schwartz, and M. W. Davidson, "Introduction to STORM." `http://www.microscopyu.com/articles/superresolution/stormintro.html`. Accessed June 27, 2013.

[14] S. Wolter, "An accurate and efficient algorithm for real-time localisation of photoswitchable fluorophores," *Bielefeld University* (diploma thesis), 2009.

[15] S. J. Holden, S. Uphoff, and A. N. Kapanidis, "DAOSTORM: an algorithm for high-density super-resolution microscopy," *Nature Methods* **8**(4), pp. 279–280, 2011.

[16] R. Henriques, E. Fornasiero, F. Valtorta, C. Zimmer, M. Mhlanga, and M. Lelek, "QuickPALM: 3D real-time photoactivation nanoscopy image processing in ImageJ," *Nature Methods* **7**(5), pp. 339–340, 2010.

[17] M. J. Mlodzianoski, J. M. Schreiner, S. P. Callahan, K. SmolkovÃ¡, A. DlaskovÃ¡, J. SantorovÃ¡, P. Jezek, and J. Bewersdorf, "Sample drift correction in 3D fluorescence photoactivation localization microscopy," *Optics Express* **19**(16), pp. 15009–15019, 2011.

[18] R. Kornhuber and C. Schuette, "Numerics 1," *FU Berlin* (lecture notes), 2001.

[19] S. Wolter, M. Schuettpelz, M. Tscherepanow, S. Van de Linde, M. Heilemann, and M. Sauer, "Real-time computation of subdiffraction-resolution fluorescence images," *Journal of microscopy* **237**(1), pp. 12–22, 2010.

[20] Adobe Developers Association, "TIFF 6.0 specification." `http://partners.adobe.com/public/developer/en/tiff/TIFF6.pdf`, 1992.

[21] J. D. Larkin and P. R. Cook, "Maximum precision closed-form solution for localizing diffraction-limited spots in noisy images," *Optics express* **20**(16), pp. 18478–18493, 2012.